



D 2.2 Dogana Metrics for the Evaluation of the Existing Tools

Work Package:	WP2		
Lead partner:	Consorzio Nazionale Interuniversitario per le Telecomunicazioni – CNIT		
Author(s):	Alessio Mulas (CNIT), Davide Ariu (CNIT), Matteo Mauri (CNIT), Enrico Frumento (CEFRIEL), Roberto Puricelli (CEFRIEL), Vincenzo Napolitano (ENG), Isidoros Monogioudis (HMOD), Bernardo Pacheco (INOV), Carlo Dambra (PROPRS), Angelo Consoli (SUPSI), Davide Andreoletti (SUPSI), Yves Mabilia (TCS), Thomas Delavallade (TCS)		
Submission date:	04/03/2016		
Version number:	2.0	Status:	Final
Grant Agreement N°:	653618		
Project Acronym:	DOGANA		
Project Title:	Advanced Social Engineering and Vulnerability Assessment Framework		
Call identifier:	H2020-DS-06-2014-1		
Instrument:	IA		
Thematic Priority:	Trustworthy ICT		
Start date of the project:	September 1 st , 2015		
Duration:	36 months		

Dissemination Level	
PU: Public	✓
PP: Restricted to other programme participants (including the Commission)	
RE: Restricted to a group specified by the consortium (including the Commission)	
CO: Confidential, only for members of the consortium (including the Commission)	

Revision History

Revision	Date	Who	Description
0.1	October 2, 2015	Davide Ariu/CNIT	First version of the table of contents and sections assigned to the partners
0.2	February 8, 2016	Alessio Mulas/CNIT	Second version of the table of contents, added a template for some sections assigned to the partners
0.3	February 12, 2016	Alessio Mulas/CNIT Enrico Frumento, Roberto Puricelli/CEFRIEL Carlo Dambra/PROPRS	Third version. Chapter1 is almost complete, Chapter 2 needs a last revision. Chapter 6 at 50%
0.4	February 22, 2016	Alessio Mulas/CNIT	Fourth version. Chapters completed. Missing only some reviews and part of chapter 6
0.5	February 24, 2016	Alessio Mulas/CNIT	Fifth version. Chapters completed.
0.6	March 01, 2016	Alessio Mulas, Matteo Mauri/CNIT	Revision
1.0	March 04, 2016	Alessio Mulas, Davide Ariu, Matteo Mauri/CNIT	Final version
2.0	November 15, 2016	Alessio Mulas	Executive Summary and Conclusions added to the previous version.

Quality Control

Role	Date	Who	Approved/Comment

Disclaimer:

This document has been produced in the context of the DOGANA Project. The DOGANA project is part of the European Community's Horizon 2020 Program for research and development and is as such funded by the European Commission. All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability with respect to this document, which is merely representing the authors' view.

Table of Contents

1.	Introduction and Background	8
1.1.	Executive Summary	8
1.2.	An overview of D2.2.....	10
1.3.	Introduction	12
1.4.	DOGANA components and evaluation.....	13
1.4.1.	Technical Definitions	13
1.4.2.	Toolchain families in detail.....	16
1.5.	Software Evaluation within the DOGANA framework	17
1.5.1.	The importance of software evaluation and its inherent difficulties	17
1.5.2.	Five critical points for Software Evaluation within DOGANA.....	18
1.6.	More on the several evaluation scenarios.....	21
1.7.	General purpose metrics for software evaluation.....	21
2.	Information gathering and analysis services.....	24
2.1.	The current Scenario.....	24
2.2.	Important elements to take into account during the evaluation	25
2.3.	Suggested Metrics and Evaluation Parameters	26
3.	Tools for the attack and hook preparation	28
3.1.	The current Scenario.....	28
3.2.	Important elements to take into account during the evaluation	28
3.3.	Suggested Metrics and Evaluation Parameters	29
4.	Tools for the Execution of the attack	31
4.1.	The current Scenario.....	31
4.2.	Important elements to take into account during the evaluation	31
4.3.	Suggested Metrics and Evaluation Parameters	32
5.	Tools for the Information Aggregation and Reporting	33
5.1.	The current Scenario.....	33
5.2.	Important elements to take into account during the evaluation	33
5.3.	Suggested Metrics and Evaluation Parameters	34
6.	Tool Evaluation Methodology	36
6.1.	Evaluation method overview and requirements	36
6.2.	Metrics definition & complete list	36
6.2.1.	Weighted metrics & weighted macro-groups.....	37
6.2.2.	Defining the general macro-group	38
6.2.3.	Defining the technical macro-groups.....	39
6.3.	Score system definition.....	41
6.4.	Performing the evaluation	42
6.4.1.	Steps to be performed prior to the evaluation phase	42
6.4.2.	Steps to be performed during the evaluation phase	43
6.4.3.	Steps to be performed after the evaluation phase.....	43
6.5.	An example of report for the evaluation of two tools.....	43
7.	Software Licenses	46

8.	Awareness Solutions	50
8.1.	Conceptual model	51
8.2.	List of possible metrics which measure awareness effectivity	53
8.3.	DOGANA metrics	54
9.	Conclusions	55
10.	Annex I: Technology Readiness Metrics (TRL)	57
10.1.	Introduction	57
10.2.	The NASA TRL levels	57
10.3.	DHS Science and Technology Readiness Level	58
10.4.	European Commission Technology Readiness Level	64
10.5.	ISO 16920 Technology Readiness Level	64
10.6.	CECOM Technology Readiness Level (TRL)	65
10.7.	U.S. Army Software Readiness Level (SRL)	68
11.	Annex II: D2.2 Checklist for theoretical assessment deliverables	71
12.	References	74

List of figures

Figure 1 – Simplified view of the role of SDVA as a retrofitting tool, used to improve performance of awareness programs and ultimately to reduce risk.....	51
Figure 2 – A model of the information space as a business process where the user, attending the awareness program, has an impact on the information systems	52
Figure 3 – The DHS TRL calculator interface	63

List of Tables

Table 1 – A list of the several evaluation scenarios and the actors involved	20
Table 2 – Software Sustainability Institute list of evaluation criteria	22
Table 3 – Macro-group overview	37
Table 4 – Macro-group “general” metrics.....	38
Table 5 – Macro-group “IGAS” metrics	39
Table 6 – Macro-group “TAHP” metrics	39
Table 7 – Macro-group “TEAT” metrics.....	40
Table 8 – Macro-group “TIAR” metrics	40
Table 9 – Score system possible values with definition	41
Table 10 – Evaluation example – complete list of metrics.....	44
Table 11 – Evaluation example – macro-group “general” scores.....	44
Table 12 – Evaluation example – macro-group “TIAR” scores.....	45
Table 13 – Evaluation example – final scores	45
Table 14 – Licenses' overview	46
Table 15 – Notes on the licenses.....	48
Table 16 – NASA TRL levels.....	58
Table 17 – DHS modified TRL	59
Table 18 – DHS TRL questions	59
Table 19 – Technology Readiness Level (TRL)	64
Table 20 – ISO 16290 Technology Readiness Level (TRL).....	65
Table 21 – Technology Readiness Level (TRL)	66
Table 22 – U.S. Army SRL.....	68
Table 23 – D2.2 Checklist for theoretical assessment deliverables	71

Definitions and acronyms

IGAS	Information gathering analysis services. One of the tool-families that are part of DOGANA toolchain
SDVA	Social Driven Vulnerability Assessment
TAHP	Tools for the attack and hook preparation. One of the tool-families that are part of DOGANA toolchain
TEAT	Tools for the execution of the attack. One of the tool-families that are part of DOGANA toolchain
TIAR	Tools for the information aggregation and reporting. One of the tool-families that are part of DOGANA toolchain
TRL	Technology Readiness Level

1. Introduction and Background

1.1. Executive Summary

The DOGANA project aims to design and to develop a framework for Social Driven Vulnerability Assessment (SDVA). The final goal is to provide organizations and companies with a set of tools in order to help them assess their vulnerability to Social Engineering-based attacks. DOGANA has been imagined as a framework that combines third-part open sources software and Web Services with new software components. DOGANA does not focus just on performing vulnerability assessment but also on awareness campaigns whose level of success needs to be evaluated.

All evaluation scenarios have been taken into account by the consortium (i.e., software evaluation, awareness campaign). For each scenario, key problems have been identified and specific solutions and guidelines has been proposed.

The number of software that can be considered potential candidates for integration within DOGANA is huge. On one hand there is the need to find and select only the best tools while, on the other hand there is the complexity of evaluating a piece of software using a scientific approach. This issue applies not just to third party tools but also to software developed within the project. In this regard, DOGANA consortium analysed all inherent difficulties in a software evaluation phase focusing on specific families of software that are deemed of interest for DOGANA.

A successful awareness campaign is very important in order to reduce the vulnerability of tested companies and organizations. Measuring the success of these campaigns is not an easy task but is a necessary step nonetheless in order to ensure that further awareness campaigns are designed and performed in the best possible way.

This document belongs to DOGANA WP2 'Framework specification and design', describes both the results and approach used to achieve the previously listed results. This document paves the way for the actual tool evaluation phase whose organization and results are described in D3.1.

In order to clarify the purpose and scope of "evaluation" within DOGANA, this document describes in Chapter 1 several different evaluation scenarios: software evaluation (i.e., third party tools and custom developed tools), Technological Readiness Level evaluation (i.e., referring to the whole project), Awareness Campaigns success (i.e., performed using DOGANA). In regards of software evaluation, the consortium analysed the problem and highlighted five critical points as especially relevant for DOGANA. For each problem, a set of

guidelines and solutions are proposed. The chapter also classifies third party tools as belonging into “tool-families” according to their role within the framework: tools for information gathering and analysis services, for attack and hook preparation, for attack execution and finally tools for information aggregation and reporting. Finally, the chapter introduces a unified methodology and a set of metrics to measure software quality (i.e., third party and custom developed) and the level of success of Awareness Campaigns. These last topics are here briefly introduced and described in details in the next chapters.

Understanding trends, recurring patterns and key elements is fundamental in order to evaluate software, this is especially true in the case of web-oriented tools such as those belonging to the information gathering and analysis services tool-family. Chapter 2 describes the current scenario, highlights important elements that should be taken into account during the evaluation phase (e.g., API stability, technical limitations, legal limitations) and also proposes a list of metrics of practical use to evaluate this family of tools.

Third party tools that can be used to prepare “attack and hooks” required for the Social Driven Vulnerability Assessment can be classified as belonging to two categories: tool that perform a single task or complete stand-alone suites. Chapter 3 analyses both cases, highlights pros and cons of both possible choices and presents a set of metrics that can help the evaluators in selecting the right tools for DOGANA.

The analysis of currently available tools has also focused on those required to actually perform an attack. The main outcome of this research, presented in Chapter 4, is an overview of how modularity, attack customization and automation can be considered as important factors. Also a list of evaluation metrics is suggested.

Data aggregation and reporting, including data visualization, are a key element in showing the results of a Social Driven Vulnerability Assessment, therefore is very important to know what are the major trends and challenges when it comes to find the proper tools and technologies to perform this role within DOGANA. The result of the research performed is presented in Chapter 5 and, as already seen in the previous chapters, offers some guidelines and evaluation metrics both for Data Aggregation and for Reporting.

Starting from the results of the previous analysis on currently available software and guidelines on how to evaluate them, the consortium developed a methodology for the measurement of several software “qualities” in order to select, among all candidates, which tool to integrate within DOGANA. The result of this work is a sound method based on a set of evaluation metrics, divided into five groups, a score system and a set of templates useful during the actual evaluation phase. All this work is presented in Chapter 6 and includes a full

example of use and a step by step guide. Actual software evaluation is out of the scope of this document, please refer to D3.1 for more information on the subject.

The DOGANA framework design has to take into account several elements such as limitations regarding the license of the final product. Since software licenses of third-party tools integrated within DOGANA have a huge impact on the overall license of the final product, the consortium has taken great care into analysing all currently available software licenses and consider which are more suitable for DOGANA. The result of this research is presented in Chapter 7.

DOGANA aims at helping organizations not just in performing vulnerability assessment but also in regards of awareness campaigns in order to actually reduce the assessed vulnerability.

Measuring the success of an Awareness Campaign is a very important step in the process. The result of a research on this subject, presented in Chapter 8, shows that while evaluation metrics are an invaluable tool a different methodology is required. Awareness campaign success and software quality evaluation are different scenarios; this requires a different approach for the evaluators.

In regards of awareness campaign success evaluation, this chapter proposes two different strategies. The first one is actually based on a two levels assessment: a first level measures the impact of an awareness campaign at user level while a second measures the impact at enterprise level. Both measurements are required to evaluate the success of the campaign. The second strategy aims at measuring the overall robustness of an enterprise protection system composed of a human and a technological part.

The consortium has also performed a research on the currently used methodologies to measure the Technology Readiness Level. Although beyond the scope of task T2.2, and with no direct impact on the tasks that will benefit from this deliverable (T3.1 and T6.1), the result of this research is deemed as useful to start paving the way toward the evaluation of the whole DOGANA framework. The result of this work is a set of tables that will guide through the evaluation of the TRL and is presented as Annex I.

1.2. An overview of D2.2.

The aim of this document is to provide a unified methodology and a set of metrics to evaluate Software Quality and Awareness Campaigns level of success.

This document starts with some technical definitions and by clarifying some concepts. Problems and issues typically faced during Software Evaluation are then analyzed with a special attention to those of interest for DOGANA. General guidelines on how to tackle the

inherent difficulties in software evaluation are provided and in the following chapters (see chapters from 2 to 5) all current software families of interest are analyzed in detail providing a list of suggested metrics and pointing out the key factors that must be considered during the evaluation phase.

Finally, an evaluation method is proposed that, albeit complete and relatively fast and simple, is provided as a highly customizable template that can be tailored to satisfy the practical needs of those that are going to perform the evaluation phase.

Chapter 7 compares several kinds of software licenses highlighting pros and cons of each of them and suggesting a list of licenses of interest for DOGANA.

The next chapter focuses on providing specific guidelines on how to perform an evaluation of the success of an Awareness Campaign performed using DOGANA.

After a brief chapter about the conclusions reached within this document, there is a final additional “annex” chapter that compares several approaches to the evaluation of Technology Readiness Levels (TRL onward in the document). This chapter is meant to be helpful during the final evaluation of DOGANA.

This is the general structure of this deliverable:

Chapter 1 – describes general problems in regard to software evaluation and how they might affect DOGANA. Highlights possible solutions and describes the general approach used in DOGANA to evaluate software and Awareness Campaigns.

Chapters from 2 to 5 – they are focused on particular family tools, they describe the current scenario for that family of software and provide specific metrics.

Chapter 6 – detailed description of the software evaluation method proposed for DOGANA. Includes the complete list of metrics, score system and evaluation sheets examples.

Chapter 7 – describes what license types are compatible with DOGANA and how they impact which software can be chosen.

Chapter 8 – it is focused on Security Awareness Campaigns, describes the scenario and provides metrics for this kind of evaluation.

Chapter 9 – it is focused on shortly present the results of the entire document.

Chapter 10 (Annex I) – it's focused on Technology Readiness Level, provides definitions, example of metrics and guidelines on the evaluation of TRL.

Chapter 11 (Annex II) – D2.2 Checklist for theoretical assessment deliverable

Chapter 12 – references.

1.3. Introduction

DOGANA will use a huge number of tools and software, some of them are complete and mature, others are simple scripts. The tools will be integrated in a complex framework.

DOGANA is a framework that combines together third-party open source software, scripts and Web Services with new code whose main purpose is to integrate together the several components and fill the gaps between them, and adding new functionalities.

DOGANA is a framework, it shares with other frameworks some classical features:

- **Inversion of control:** in DOGANA, unlike in normal software collection or libraries, the overall workflow is not dictated by the user, but by the framework.
- **Default behavior:** DOGANA, as every framework, has a default behavior.
- **Extensibility:** DOGANA can be extended by adding new components to the tool-chain.
- **Non-modifiable code:** DOGANA own code is not supposed to be modified. Users can extend the framework but should not modify its code.

These characteristics of frameworks must be kept in mind during the evaluation of what are the best tools to be included in DOGANA: tools whose purpose is clearly defined can easily become part of a workflow. Solid, well documented API and interfaces make extending the framework easier and improve general re-usability of the code.

An ideal candidate to become part of DOGANA toolchain must be carefully picked among a huge number of available software solutions, Web Services, scripts, etc.

On one hand there is the need to find and select only the best tools while, on the other hand there is a huge number of tools to evaluate in a limited amount of time while using a scientific approach. The answer to this problem may lie in the words of two scientists of the past.

As the famous French chemist and biologist Louis Pasteur once said *“a science is as mature as its measurement tools”*. Evaluating software must be done with a scientific approach in mind, using the appropriate tools. It's worth quoting also the British mathematician and philosopher Bertrand Russell who said that *“although this may seem a paradox, all science is based on the idea or approximation. If a man tells you he knows a thing exactly, then you can be safe in inferring that you are speaking to an inexact man”*. The right tools are those that allow us to evaluate something with a well-known error.

This deliverable provides a unified methodology for metrics that are going to be used all along the project development to find the right tools, those with the “exact amount of approximation”. This deliverable will also provide guidelines and metrics to evaluate DOGANA as a whole from the point of view of Technology Readiness Level and also to measure the success of Awareness Campaigns performed using DOGANA.

1.4. DOGANA components and evaluation

This paragraph contains some technical definitions that will be used in this deliverable and all along the project. A short list of technical terms will be followed by a description of all the external family of tools that will be included in DOGANA.

1.4.1. Technical Definitions

This paragraph contains some technical definitions that will be used in this deliverable and all along the project.

API – a set of protocols, routines and tools for building software and applications. The purpose of an API is to express the functionalities of a software in terms of interface (e.g., operations, inputs, outputs, underlying types, etc.) allowing for changes in implementation without compromising the interface itself. API can be a simple specification of remote calls exposed to consumers (e.g., SOAP and REST services), a library (e.g., specifications for routines, classes, variables, etc.), a set of classes with associated list of class methods (e.g., documentation of all the kinds of objects one can derive from the class definitions, and their associated possible behaviors).

Note: API are associated to both frameworks and libraries. A certain behavior can be implemented by a library or built into a framework, in either cases it's described by an API.

Attack Surface – according to Open Web Application Security Project (OWASP): *“the attack surface of a software environment is the sum of the different points (the “attack vectors”) where an unauthorized user (the “attacker”) can try to enter data to or extract data from an environment”*. [1]

Attack Vector – According to TechTarget: *“An attack vector is a path or means by which a hacker (or cracker) can gain access to a computer or network server in order to deliver a payload or malicious outcome”* [2]

Framework – a universal and reusable software environment that provides a particular functionality supposed to be part of a larger software application. The purpose of a framework is to ease the development of software applications, products and solutions. Frameworks provide a general solution that can be made more specific by the end-user by adding some user-written code.

Note: DOGANA is a framework and it's possible that it will contain other frameworks within itself.

Library – a collection of software resources (e.g., previously written code, classes, configuration data, etc.) supposed to be used by computer programs. The purpose of a library is to encourage the sharing of code, to increase the modularity and ease the distribution. Libraries provide a collection of implementations of behavior, each with a well-defined interface used to invoke it. A program can invoke a library and get the behavior implemented by it without any need to be implement it again.

Mashup – software architecture used in web development based on a web server that consumes several Web Services hosted on different machines and combines them into a single user interface. The purpose of a mashup is to provide easy and fast access to content from several sources by a unified interface in the form of a web application. The main characteristics of a mashup are: aggregation, combination and visualization. There are many types of mashup, the most common being: Business/Enterprise (internal resources are combined with external ones into a visually rich web application), Consumer (combines mixed kind of data from several public sources into a well-organized interface in the web browser), Data (combines similar types of media and information from multiple sources into a single representation).

Payload – in computer security, the word payload refers to the part of a malware that actually performs a malicious action. In telecommunications it usually refers to the part of transmitted data that contains the “message” sent; payload doesn't include any headers or metadata, this part is generally referred to as overhead data.

Quality – according to ISO/IECⁱ the definition of Quality is the “*capability of a software product to conform to requirements*”.

Script – this word may refer to a program written for a specific run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator (e.g., a bash script on a Unix shell, a batch file on a Windows shell, etc.) or to a small program (up to a few thousand lines of code) in high-level general purpose language (e.g., Perl, Python, etc.). Scripts must not be confused with libraries, while the later are organized to be used by multiple programs and promote code reuse, the former are, on the opposite, organized to be used as standalone software. Scripts can be parts of larger software but are not created with that purpose in mind.

Scripting language – A scripting or script language is a programming language that supports scripts. Scripting languages are often interpreted (rather than compiled), their primitives are

ⁱ ISO/IEC 9001 (International Organization for Standardization, “ISO/IEC 9001: Quality management systems – Requirements”, 1999.), commented by International Organization for Standardization, “ISO/IEC 24765: Systems and software engineering — Vocabulary”, 2010

usually elementary tasks or API calls that can be combined into more complex programs using the language. Scripting languages are also sometimes referred to as very high-level programming languages, as they operate at a high level of abstraction. Scripting languages can be used to develop frameworks and libraries and not just scripts.

Shellcode – small piece of code, usually written in machine language, that can be used as payload during the exploitation of a software vulnerability. The name shellcode derives from the fact that the purpose of this code is generally to start a command shell that the attacker can use to control the compromised machine.

Shell script – a script designed to be run by the Unix shell. All the various dialects of shell scripts are considered to be scripting languages.

Tool – a generic term that refers to a software component that is, or is a candidate to be, part of DOGANA framework. Libraries and frameworks, although software components, are not considered tools because they can't be used as stand-alone software but require some “main code” to invoke them.

Toolchain – a group of tools that can be combined together to accomplish a task. DOGANA toolchain is composed on a case by case base with tools that belong to the framework.

Tool-family – high-level abstraction used in DOGANA to describe tools that belong to its framework. Every tool is supposed to be part of one of four families that share a purpose/role in the toolchain. The four tool-families are: information gathering and analysis services (also referred as IGAS), tools for the attack and hook preparation (TAHP), tools for the execution of the attack (TEAT), tools for the information aggregation and reporting (TIAR).

Web Service – It is a software function provided across the web according to the concept of utility computing. A Web Service is a communication between two electronic devices over a network using a protocol such as the HTTP and based on the exchange of machine readable files in formats such as XML and JSON. Web Services are divided into two major classes: Arbitrary Web Services (the service expose an arbitrary set of operations) and REST-compliant services (the service manipulate representations of web resources using a uniform set of stateless operations).

Note: Web Service term is often substitute with SOAP or REST, these two terms indicate the exact technological approach for Web Service implementation.

Workflow – high-level description of the use of DOGANA framework that can be depicted as a sequence of phases, each of them requiring the use of one or more tools. The entire

sequence of tools used during the workflow is called toolchain. For each phase of the workflow there is a collection of tools that can be used, this collection is called tool-family.

1.4.2. Toolchain families in detail

Every tool that is going to be part of DOGANA framework has to be classified into one of these four categories known as “tool-families”. Each one of these categories has a special purpose in the toolchain, is used in a specific phase and has a “role”. These are the four tool-families explained in detail:

Information Gathering and Analysis Services (IGAS)

Purpose of this phase – to do some research on the target and collect enough information to build a successful hook.

Purpose of the tools – harvest information from several sources, collect and organize the information to allow the attacker to perform searches and analysis on it.

Example of tools – online search engines, data mining scripts, etc...

Tools for the Attack and Hook Preparation (TAHP)

Purpose of this phase – to set things up for a successful attack, create a scenario and build the trust with several elements (pretexting, fake websites).

Purpose of the tools – help during the attack planning (selection of the best target, including possible strategies and identification of psychological levers), help during the scenario creation (pretexting, creation of fake website, fake profiles, creation of phishing emails, chat bots, etc.).

Example of tools – Maltego Case Fileⁱⁱ, SETⁱⁱⁱ, automated website cloning, DNS spoofing tools.

Tools for the Execution of the Attack (TEAT)

Purpose of this phase – to maintain the charade and strengthen the control of the relationship long enough to extract the information and, optionally, close iteration without arousing suspicion. Create the actual attack vector (i.e., attach a malware to a file like a PDF, docx, etc.).

Purpose of the tools – creation of the actual attack vector by combining a malware with a premade document (prepared during the previous phase), by creating some "interesting software" (e.g., fake patch/update for well-known software, infected fake free software, etc.)

ⁱⁱ <https://www.paterva.com/web6/products/casefile.php>

ⁱⁱⁱ <http://www.social-engineer.org/framework/se-tools/computer-based/social-engineer-toolkit-set/>

or by setting up some remote attack tool that can work once the victim as visited a link. Tools that increase the chances of success of the attack by obfuscating the malicious code or altering it to avoid any Antivirus available to the victim. Tools that can help maintaining the charade: proxies, ambient sound generators or audio files (e.g., vishing), automatic message writer for social network (e.g., to plan interaction at scheduled times).

Example of tools – Metasploit^{iv}, Core Impact^v, Canvas^{vi}.

Tools for the Information Aggregation and Reporting (TIAR)

Purpose of this phase – to organize the collected data and extract only the useful information and to write down an attack report.

Purpose of the tools – to collect and store a large amount of data of different formats (e.g., text, images, sounds, captured data traffic, etc.), to automatically generate full or partial reports on the attack providing the selected amount of information. To generate graphs and tables. The reports must be available in different formats.

Example of tools – Tools that convert documents from one format to another one, tools that create graphics in formats such SVG starting from raw numerical data.

1.5. Software Evaluation within the DOGANA framework

1.5.1. *The importance of software evaluation and its inherent difficulties*

Software Evaluation refers to a process of gathering data and then analyzing it in order to be able to determine whether the software is achieving its stated objective and anticipated results.

Software Quality is defined as “the capability of a software product to conform to requirements”. There is clearly a strong link between Software Evaluation and Software Quality: in order to perform the former, we have to measure the later.

Measuring Software Quality can be a daunting task, it requests resources and technical skills, nonetheless is still quite a common practice that can be motivated by at least two reasons:

- Cost Management
- Risk Management

^{iv} <http://www.metasploit.com>

^v <http://www.coresecurity.com/core-impact-pro>

^{vi} <http://www.immunitysec.com/products/canvas/index.html>

While they can be often considered two opposing forces, Cost Management and Risk Management sometimes just seem to be just two sides of the same coin. They can push toward increasing system performance, avoid system failures or other more security oriented parameters like avoiding data corruption or preventing security breaches.

There are at least 3 critical points that we should consider in regards to Software Evaluation:

Cost Management and Risk Management – evaluating software requires resources and this can be seen as a cost. Evaluating software can decrease risks and increase the general quality of the software, this reduces losses and the cost for risk mitigation. There are several factors to be considered (e.g., the cost of Software Evaluation, the nature of risks involved, risk mitigation's cost, etc.), no single one can be labeled as “the most important” and there must be some kind of trade-off.

Software Architecture – complex software architectures (e.g., multi-layer architecture, mashup, etc.), wide use of external libraries and/or frameworks can highly increase the complexity of Software Evaluation.

Objective Evaluation – assessing the quality of software can be tricky, balancing the subjective individual experience with sound evidence and objectivity is not easy.

All these points highlight the importance of performing quality analysis and measurement in a comprehensive and consistent manner.

1.5.2. Five critical points for Software Evaluation within DOGANA

The previous paragraph highlights three general critical points that must be faced while planning and executing software evaluation. Two more points can be added to the list to better address possible sources of confusion and difficulties: what is the software that is going to be evaluated and class of users taken into account.

The purpose of this paragraph is to analyze the impact of these five points on software evaluation performed within DOGANA. For each point a solution will be proposed to mitigate, and hopefully solve, all the difficulties described.

DOGANA's Cost and Risk management

Problem – Risk and Cost management is a complex topic, for the scope of this deliverable it should be enough considering that DOGANA is going to handle sensible data about companies and people, this requires a careful analysis of all the software components involved. How is data stored, managed and transferred? What kind of safety and privacy procedures are used? These are the kind of questions that need to be answered.

Solution – in order to guarantee a careful and safe handle of data by every software component of the framework, the evaluation method will contain one or more metrics to measure how easy is to track how the data is handled by the software (e.g., is it stored, transferred or changed? How?). Well written source code and good documentation are an example of a good trait to be look for in a software. Those involved in Cost and Risk Management will be able to quickly understand what parts of the framework are the more sensible ones.

DOGANA'S software architecture

Problem – this deliverable is not about DOGANA software architecture but some main assumptions can be made: it will have layers, there will several presentation formats (GUI, CLI, etc.), there will probably some Services (RESTful API, etc.), several Business core scripts and software not to mention different approaches to data handling (SQL DB vs non-relational ones, CSV local file storage, cloud, etc.).

Solution – a metric will measure the attitude of every software component to become part of DOGANA architecture. Several technical characteristics can be easily checked to evaluate this metric (e.g., the software has an API, is developed according to the OO paradigm, etc.).

Objective evaluation

Problem – who is going to perform the evaluation? Is the evaluation going to be done sequentially on all the software during a single step or is it going to be performed during several steps, each of them focusing on a smaller part of all the software? The objectiveness of the evaluation must be guaranteed with an ad hoc methodology.

Solution – a metric based evaluation method and a score system with a limited number of well-defined scores “steps” will increase the objectiveness of the evaluation and reduce the complexity.

Software to be evaluated

Problem – what software are we going to evaluate? The third party components and tools that are going to be included, the custom developed code that adds functionalities and glue together all the components or the final product?

Solution – as previously stated, DOGANA is a complex framework and there are several different “evaluation scenarios”. Different custom tailored approaches will be used on a case by case basis, this will reduce the complexity and duration of the evaluation phase. Metric based evaluations will be used while evaluating third party software components and to measure the success of Awareness Campaigns.

Class of users

Problem – assessment is usually performed considering a specific class of users that are going to use the software. Who is our class of users? We have the developers and the final users. The first ones are going to be involved in developing DOGANA; they will develop new tools and/or change the included ones. The second ones, the final users, are those that are going to use the final product, DOGANA framework, to perform SE assessments.

Solution – even in this case there are different “evaluation scenarios” with different Class of Users.

Table 1 – A list of the several evaluation scenarios and the actors involved

Evaluation Scenario	Evaluation performed by	Target Class of Users	Evaluation Scenario Description
Third Party tools	Developers	Developers, Final Users	Developers evaluate third party tools to decide which ones are better suited to integrate within the framework and which ones are the best ones for Final Users.
Custom developed tools	Developers	Final Users	Developers evaluate new code created by partners of the project to guarantee that it is made according to the desired TRL.
DOGANA framework	Developers	Final Users	Developers evaluate DOGANA project to measure it's final TRL
Awareness Campaigns	Developers, Final Users	Final Users	Developers and Final Users measure the success of an Awareness Campaign performed using DOGANA framework.

All the previously described critical points can be addressed by a custom developed software evaluation method that needs to:

- consider all the requirements to guarantee a careful Cost and Risk Management
- take into account the DOGANA scenario with all its distinctive characteristics (e.g., different users, etc.)
- provide tools to ease the integration work into the framework
- guarantee an objective but short evaluation phase

1.6. More on the several evaluation scenarios

As previously stated, evaluation in DOGANA is done in several different “scenarios”, each with its differences and special cases. This deliverable contains guidelines about all of them. The following list suggest guides to the right part of the deliverable according to the kind of evaluation that needs to be performed:

- **Evaluating a third party tool:** Chapter 6 contains a description of the evaluation method while chapters 2 to 5 contain detailed information on the evaluation scenario and caveats regarding each tool-family.
- **Evaluating the results of an Awareness Campaign done using DOGANA:** Chapter 8 contains all the necessary information about it.
- **Evaluating DOGANA TRL:** Annex I contains all the information on Technology Readiness Level evaluation.

It is worth mentioning that while chapters 8 and Annex I contain evaluation guidelines, chapter 6 contains an evaluation method. The reason of this choice is that third party tool evaluation requires a rigid approach to guarantee a consistent evaluation without compromising the speed of the process while both Awareness Campaign Success and TRL evaluation require a more flexible one. A strong emphasis on a purely numerical evaluation can be a source of problems during Awareness Campaign Success and TRL evaluation and better results can be achieved using broad definitions such as those presented in Chapter 8 and Annex I.

It's also important to remember that the evaluation method proposed in chapter 6, as already stated in the very begin of this document (see 1. Introduction and Background), should be considered as a *highly customizable template that can be tailored to satisfy the practical needs of those that are going to perform the evaluation phase*.

Every element of the evaluation method, from the score system to the number of metrics up to the weights assigned to them, can be easily changed to reduce or increase the required level of detail.

1.7. General purpose metrics for software evaluation

Several different approaches can be used to evaluate a software. Some are based on surveys, others on benchmarks and practical tests, others on “user-stories” and interviews. There are open source evaluation methods, custom made ones and even proprietary ones.

DOGANA suggested evaluation method, described in details in chapter 6, is a criteria-based one and provides a quantitative measure of software's quality. Assessment is made by checking whether the software exhibits or not various qualities, the more they are satisfied, the better the software is. Qualities are called “metrics” and are collected in two groups: general and technical. Each metric has its weight, and it's rated with a limited set of possible “marks”.

A criteria-based assessment is useful to take high-level decisions because it gives a measurement of quality in a number of areas [3]. It provides a consistent way to measure “quality” while reducing the complexity; metrics can be defined for each family of software that needs to be evaluated. Even the scoring system helps in reducing the complexity, a limited set of possible scores couple with a good definition of the meaning of the score itself, can guarantee that votes are going to be based on sound reason instead of pure personal opinion.

The Software Sustainability Institute [4] proposes a general set of metrics for software evaluation based on ISO/IEC 9126-1 Software Engineering – Product quality [5]. These metrics are grouped together in “areas”. This is the original list:

Table 2 – Software Sustainability Institute list of evaluation criteria

Criterion	Sub-criterion	To what extent is/does the software...
Usability	Understandability	Easily understood?
	Documentation	Comprehensive, appropriate, well-structured user documentation?
	Buildability	Straightforward to build on a supported system?
	Installability	Straightforward to install on a supported system?
	Learnability	Easy to learn how to use its functions?
Sustainability and maintainability	Identity	Project/software identity is clear and unique?
	Copyright	Easy to see who owns the project/software?
	Licensing	Adoption of appropriate license?
	Governance	Easy to understand how the project is run and the development of the software managed?
	Community	Evidence of current/future community?
	Accessibility	Evidence of current/future ability to download?
	Testability	Easy to test correctness of source code?
	Portability	Usable on multiple platforms?
	Supportability	Evidence of current/future developer support?
	Analysability	Easy to understand at the source level?
	Changeability	Easy to modify and contribute changes to the developers?
	Evolvability	Evidence of current/future development?

	Interoperability	Interoperable with other required/related software?
--	------------------	---

DOGANA metrics, as previously stated, are divided into two groups: general and technical.

The first group will be based on general metrics, like those from the Software Sustainability Institute. These metrics will help developers in measuring how well each tool can fit in the framework, how easy is to modify it and change it but also how easy it will be for the final user to learn it. It's easy to notice that these metrics will be more useful in designing the framework and in the integration part of the project rather than in finding which tool is the best at “doing its job”. That part of the evaluation will be based on the second group of metrics: technical. For more information on the evaluation method and a complete list of the metrics involved please read chapters 2, 3, 4, 5 and 6.

2. Information gathering and analysis services

The aim of this chapter is to support the “Information gathering and analysis services” tool selection process. It starts by describing the current scenario and follows by pointing out the main elements to take into account and presenting a list of proposed metrics.

2.1. The current Scenario

Information gathering and analysis services tools belong to the first tool “family”. As a first step in DOGANA toolchain, this family has the duty of harvesting and collecting data from a source and, when possible, to provide a first analysis of this data in order to support the activities of the tools that come next in the toolchain.

What is a *source of data* from the point of view of this chapter? The concept of *source of data* and that of *attack surface* are strictly linked together. In chapter 1 attack surface is defined as:

The attack surface is the sum of the different points where an unauthorized user can try to enter data to or extract data from an environment.

Since we're talking about data sources, let's consider for a moment a more specific definition of attack surface, one that focuses only on data extraction.

The attack surface is the sum of the different points where an unauthorized user can try to extract data from an environment.

Since single users, companies and associations are all potential targets for SE based attacks then it's pretty straightforward to consider all the data they *store* on the web, whether they do it deliberately or not, as a valid attack surface.

As a consequence, a source of data is basically the entire attack surface of the target, the footprints of its interaction with every social network, online community and the web itself.

An incredible number of data sources exists: from well-known social networks to forums and online communities. All sources naturally change during their “lifetime”, not just in their content (e.g., a text-only community decide to start storing images too) but also in size (i.e., the community becomes bigger or smaller).

Different sources can merge together into a new source or split and separate; this process usually happens when two IT companies merge together, or a larger one acquires one or more smaller ones. As an example of interests we can cite two famous acquisitions by the former Google Inc. (now Alphabet Inc.): Picasa (July 13 2004) [6] and YouTube (October 9 2006) [7][8].

Information sources can also disappear for several reasons: a social network user base is so small that it's not worth keeping it alive (i.e., the list of “dead” social networks is quite impressive [9]. Sometimes the reasons behind the failure of this projects are deeply analyzed for the improvement of future projects [10]), a Web app that harvest data from a social network can't do it anymore because of new API rules, etc.

Every source usually requires some custom tool to be harvested, especially if no official API has been released. Data sources and harvesting tool are bind together in such a way that even a small change in the data source's code usually requires substantial changes in the tool's code.

Given the huge number of sources of information, as a logic consequence, it's impossible to know the exact number of tools that belong to the “information gathering analysis services”, even giving a rough estimate is quite a hard task due to the ever changing nature of the sources of information.

Tools range from complex software (e.g., desktop apps with a GUI, web app, etc.) to simple scripts or libraries available for the most used programming languages. Some tools are specialized in one source (e.g., a single social network) while others can perform broader researches on several sources but often with inferior result compared to the more specialized ones.

New harvesting tools are created almost every week but they usually become obsolete in a very short time, either because of some change in the source of information or because a better tool is created. Tools based on API usually are less prone to this problem compared to tools that harvest information straight from HTML pages. API changes are less common than changes in the layout of a web page.

Tools that belong to this family are usually simple scripts or small libraries for a common scripting language (e.g., Python, Perl, etc.) written by a single developer. Sometimes these tools can be used from a web based interface like a web app or a single application, in rarer cases they are services exposed to the web via an API of some sort (RESTful ones being the most common).

2.2. Important elements to take into account during the evaluation

It is worth considering several key factors while performing a software evaluation for tools that belong the Information gathering and analysis services family.

Data mining authorization – while some tools use official APIs to harvest data from their sources other tools don't. Several social networks expose part of their data via API and block, or try at least try to limit, other forms of data mining. Companies use several approaches to limit or prevent data mining, sometimes it's a legal one, other times it's strictly technical. There are several cases where data mining, especially with the aid of automated tools, is often prohibited and considered a violation of contract, this can be seen as a purely legal “block”. There are also examples of technical blocks, these are usually based on some technical aspects like logging policies, traffic analysis (e.g., a single page application that shows data only to logged users, HTTP request header analysis, cross-site domain request policy). Only legal compliant tools can become part of DOGANA. In order to respect all legal requirements and limitations, both “source's legal terms” and tool source code must be carefully checked (e.g., is an API required? Is there a daily limit on the number of queries?).

Technical limitations and parameters – data mining can require a considerable amount of software and hardware resources and can be a cause of technical problems for the target source infrastructure, this is especially true if the number of performed queries is very high or generates a high traffic load. It's worth considering what is the average number of queries that can be performed without causing technical problems, the traffic that can be generated or the frequency of the queries. Sometimes these technical limits are known, even clearly stated in the API documentation, sometimes they are not. When evaluating a tool, there are some questions that must be addressed: is the tool compliant with the official regulation? Is there a way to set a limit to the queries performed by the tool?

Output format – data format is a very important factor to increase the quality of data storage and data transmission. What format, or formats, are used by the tool to export and show the result of its queries? How many formats are supported by the tool?

Data storage – is there some data storage functionality integrated in the tool or is the result of the data mining just forwarded as output? Is there some filtering and data aggregation functionality? If there is some kind of storage, what are its characteristics? (e.g., relational db, non-relational db, file, archive of files?)

2.3. Suggested Metrics and Evaluation Parameters

This paragraph's aim is to suggest metrics that can address previously described pitfalls and help in the evaluation process. These metrics, as the others presented in the next three chapters, are only guidelines. There is no definitive metrics set on evaluating software and this list should be corrected and changed once it's clear how many tools are going to be evaluated and what are their general characteristics.

Number of sources – number of information sources like social media, documents, public web sites, blogs that the tool is capable to search for. Theoretically, the more sources a tool is able to target, the better is.

Output formats (number and kind) – Number and kind of output formats for further processing (e.g., txt, csv, XML, etc.) and existence of compatible applications/tools that can directly process tool's outcome.

System requirements & performance – minimum specific system requirements for tool optimal performance, the lesser are the better the tool is. It's worth considering that sometimes a faster tool is to be preferred to a slower one, even if its system requirements are higher. Sometimes speed can be limited due to technical limitations from the target.

Information correlation capability – existence and/or number of correlation techniques for gathered data and relevance of the achieved results.

Platform dependency – are there any specific platform/software dependencies for tool's usage?

Information relevance – how relevant is the retrieved data or generated information with the provided search criteria?

Information filtering – how good is the tool at filtering the harvested data and perform customizable queries?

3. Tools for the attack and hook preparation

The aim of this chapter is to support the “Attack and hook preparation” tool selection process. It starts by describing the current scenario and follows by pointing out the main elements to take into account and presenting a list of proposed metrics.

3.1. The current Scenario

Software that belongs to the “attack and hook preparation” family usually falls into two categories: those that perform one simple task (e.g., to send anonymous emails) and those that allow the handle the creation of a complete attack scenario (e.g., to clone a website, automatically sending fake emails, create statistics and reports about victims, etc.).

Simple scripts and Web Service are good examples of the first group while frameworks and complex software belong to the second one.

Software that belongs to the first category is generally the result of the effort of a single developer while frameworks and, generally, software of the second categories don't follow a single trend: some of them are created by a single, albeit expert, developer while others are made by companies or the result of a community group work.

While generally free and open-source, these tools sometimes provide advanced features for a small fee. Usually these tools are updated whenever the data source adopt some security countermeasure that prevent the tool to work as expected.

3.2. Important elements to take into account during the evaluation

Several key factors must be considered while evaluating the quality of software that belongs to the attack and hook preparation tool-family:

Community trustworthiness – due to the nature of this kind of tool, looking for non-official online resources (e.g., tutorials, previews, examples, etc.) about them can be difficult. The result of such researches can be misleading (e.g., review performed by an overly enthusiast non expert user), potentially dangerous (e.g., a complete guide to the tool provided as free malware infected pdf) or complicated and long (e.g., information is provided via private message on a forum). Some of these tools, albeit often completely legal, are notoriously used for nefarious purposes and because of this some of the source of information regarding these tools are shady forums, dark-web based communities, etc.

Emulation capabilities – hook preparation often requires the creation of fake entities, such as fake social network profiles, fake web sites and the like. These fake entities need to faithfully emulate their digital world counterparts. It's important to evaluate the quality of these hooks

(e.g., what is the likelihood of a victim not being able to realize that the hook is fake?). It's worth considering what is the nature of the “target”, there are cases where the fake entity must be able to avoid being detected by automated systems or technical scrutiny (e.g., hosting service spotting a fake phishing page, automated removal of fake profiles in a social network, etc.).

Customization – how many degrees of freedom are provided to the user? Is the level of customization high or low? There are cases when the user just wants to perform a simple task (e.g., cloning a web site), others when a specific result is required (e.g., creating a highly detailed fake profile on a social network). A highly customizable tool isn't necessary good; it all depends on its role in the toolchain.

Additional data harvesting capabilities – hook preparation albeit a separate phase from data harvesting one, still can be a source of additional useful information. Is the tool useful in identifying potential new targets? A tool can be able to provide statistics about how the target, and other users, are interacting with the fake created entity (e.g., a cloned website provides information about the visitors, a fake social network profile create logs and graphs about its contacts, etc.).

Adaptability – how many tasks can be performed with the tool? Is it a single purpose tool or a complete framework?

3.3. Suggested Metrics and Evaluation Parameters

Attack and hook preparation is a phase that can be performed in a thousand of different ways, each requiring a different set of skills and instruments. It's very hard, if not impossible, to find a single software tool able to cover all the possible required “roles”. Adaptability is the key. The final user is usually required to use several tools for different tasks. Four general metrics can apply to every candidate that belongs to this family:

Community Strength – how many users do use the tool? Is the community of users strong and reliable? (see section 3.2 for related problem involving this assessment).

Automation – level of automation in identifying potential targets, bypassing security challenges (e.g., captcha resolution, security questions, e-mail verification, etc...), submitting plausible comments/posts (e.g., in a social network or in a blog), following capabilities (i.e., automatically decide the people to follow in a social network), generating parameters (e.g., random names).

Level of customization – does the tool provide standard templates only? Is it possible to customize the generated resources?

Number of functionalities – this metric assesses the completeness of the tool from a qualitative (i.e., heterogeneity of its features) and a quantitative (e.g., number of possible actions) point of view. How many different functionalities are provided by the software? How many “actions” is possible to perform? Is this software able to provide enough functionalities or is it required to use it together with other tools?

The next metrics are tailored for tools whose purpose is to generate “fake entities” (e.g., fake profiles on social networks, fake users on chat rooms, etc.):

Popularity of the target platform – how popular and known is the platform where the fake entity is going to be deployed?

Number of platforms – how many social platforms (e.g., social network, chat, VoIP services, etc.) can be targeted with the evaluated tool?

Fake identity detail level – how good is the tool in creating and deploying a fake entity on the platform of interest? How easy it is to recognize a fake entity when it is in a group of real ones?

4. Tools for the Execution of the attack

The aim of this chapter is to support the “Attack execution” tool selection process. It starts by describing the current scenario and follows by pointing out the main elements to take into account and presenting a list of proposed metrics.

4.1. The current Scenario

Whatever theoretical model we frame social engineering attacks onto, it always includes one stage in which the attacker exploits the human factor, to gain foothold into some system, such that a subsequent technological attack might be carried on. Consequently, often the choice of the attack vector an attacker will use to lure the victim, and the strategy he will use to succeed at that attack, precedes the actual selection of the tools that will be supporting it. Usually he will also need different kind of tools to support a single attack. For example, on a phishing attack, where the victim is expected to open some malicious attached file, an attacker will need a tool that enable him to craft emails as well as an additional tool to generate malicious payloads to be included in the email.

As the attacker can choose from several attack vectors to convince the victim to take different kind of actions, he will need to pick its tools from a wide selection. As a reference on social engineering tools, there is SET (Social Engineering Toolkit) [12], that aggregates several attacks against a person or organization. It is an open-source toolkit created and written by David Kennedy and supported by the security community at large.

During the selection process, attacker should be aware that some tools lose their effectiveness as long their operation is known by the community and the defensive tools are improved accordingly. This is the case of exploit-based tools. This kind of tools needs a strong support and maintenance by their users’ community. The opposite occurs with the range of tools based on communication standards to spoof the caller ID, on VoIP calls or SMS, or spoof the sender address on an email. In this case there are long lasting tools, given that the defensive strategies are not as easy to implement as in the other cases.

4.2. Important elements to take into account during the evaluation

The first requirement pointed out when someone is looking for an attack tool is its effectiveness. Whoever is looking for this type of tool wants, for sure, to be successful using it and to achieve his purpose.

Unfortunately, there is not a unique metric for effectiveness. It depends on a wide range of factors. Some are inherent to each tool, while others are dependent on the attacker skills and also on the attack target. Prior to the attack tools selection process, those factors need to be translated into a set of parameters that will be evaluated by one or more metrics. The selected metrics should be comprehensive, broad, and result on a set of comparable values (e.g., yes or no, numeric or scale value).

The suggested set of metrics included on 4.3 tries to evaluate the attack tools, considering its features, usability and needed skills, robustness, maintenance and ethical reputation. The application of these metrics should also bear in mind the attacker goal (i.e., tools to be used on a single target attack should not be assessed regarding their “mass attacks level”).

4.3. Suggested Metrics and Evaluation Parameters

Multi-attack availability – evaluates the attack adaptability of the tools by measuring the number of different available attacks, the number of attack vectors, etc.

Multi-attack combination – is it possible to combine different of attacks in a sequence? How many attacks can be combined?

Automation – is it possible to automate the actions needed to perform an attack?

Mass attacks level – is it possible to organize and launch mass attack campaigns?

Exploits up-to-date – are the exploits used by the tool up-to-date, frequently updated and changeable? The frequency of the updates can be used as metric. It's worth considering that not all tools are exploit based.

Third-party exploits integration – is it possible to integrate third-party exploits or is the tool limited only to embedded exploits? If exploit integration it's possible, how easy is to add a new exploit to the tool?

Attacker's identity concealment – this metrics evaluates if the attacker identity is exposed. A possible way to measure this parameter is to look at what kind of information about the attacker is leaked during the attack phase.

Identity spoofing – evaluates the capability of the attacker to assume a fake identity. Is it possible to configure a “fake identity” that will be showed to the victim?

Persistence – evaluates if this tool can ensure a persistent access to the target.

5. Tools for the Information Aggregation and Reporting

The aim of this chapter is to support the “Information aggregation and reporting” tool selection process. It starts by describing the current scenario and follows by pointing out the main elements to take into account and presenting a list of proposed metrics.

5.1. The current Scenario

The aim of this section is to describe which metrics will be used, in the context of the DOGANA project, to evaluate the existing tools for Information Aggregation and Reporting.

These tools will be used in the context of the DOGANA framework to visualize, for instance, the overall results of the Social Vulnerability Assessment and the related information. Metrics described in this section will be then used in Task 3.1- Evaluation of the landscape and Gap Analysis for the selection of the most appropriate visualization and reporting tools for the DOGANA toolchain.

The set of visualization tools available nowadays ranges from facilitating the creation of charts and dashboards to the provisioning of advanced functionalities such as predictive and statistical analysis.

The challenges addressed by these tools includes analysis, capture, search, sharing, storage, transfer, visualization, querying and information privacy. Most of them are open source projects to create a big data fusion, analysis, and visualization platforms designed for anyone to use. Intuitive web-based interface helps users discover connections and explore relationships in their data via a suite of analytic options, including 2D and 3D graph visualizations, full-text faceted search, dynamic histograms, interactive geographic maps, and collaborative workspaces shared in real-time. As an open source project, features are evolving all the time.

Therefore, among the long list of tools out there designed for different types of information aggregation and reporting, it's important to determine which ones better fit DOGANA needs, that can be summarized as follows:

- Visualization in real-time of the ongoing attack.
- The support for real time analytics using the data that are being collected.

5.2. Important elements to take into account during the evaluation

Data Aggregation and Reporting are two directly related concepts: information is first aggregated then reported in a following stage.

Before defining the metrics that would establish some criteria to evaluate tools adequately is important to understand these concepts:

Data Aggregation – is a type of data and information mining process where data is searched, gathered and presented in a report-based, summarized format to achieve specific business objectives or processes and/or conduct human analysis. Data aggregation may be performed manually or through specialized software. In DOGANA, this implies the aggregation of the information coming from the different components of the toolchain, that may use different data formats and delivery technologies.

Report – an account or statement describing in detail an event, situation, or the like, usually as the result of observation, inquiry, etc. In DOGANA, this implies the reporting of aggregated information to different actors, (i.e., technicians, for the verification of the Social Vulnerability Assessments process, or Decision Makers, for the early understand of the vulnerability of the enterprise).

5.3. Suggested Metrics and Evaluation Parameters

The following metrics follow the assumption that the tools to be evaluated are going to be used in the context of SE and for the purpose of Data Aggregation and Reporting as previously defined:

Data Aggregation

Information structure – after aggregating all relevant information, the report should follow a certain structure (e.g., by relevance, thematically, summarized and then extended, etc.). It is not so important to use a specific structure (although it would help to follow a standardized one) but to be coherent and use always the same one in order to avoid confusions. Also, the tool should be able to grouping results so each type of grouped data has its own method of segmenting the result set. Indeed, many developers would agree that the most powerful aspect of aggregation is the ability to nest them. So it is possible define a top-level data aggregation and, inside of it, define a second-level aggregation that operates on each result set. This nesting can go as many levels deep as required.

Adaptability/Flexibility – Analyze if the tool can adapt to different programming languages. Developers can use the native JSON-over-HTTP interface or one of the several language bindings available nowadays, like Ruby, Python, PHP, Perl, .Net, Java, and JavaScript. A tool with great functionalities can become very inefficient if it cannot adapt or at least offer certain level of flexibility.

Efficiency – As mentioned in previous points SE gathers large amounts of information and tools should be fast enough to process it, but at the same time the examination should be very thorough and the results should be understandable.

Reporting

Reporting format – Analyze the different formats used to export and deliver the reports (XML, pdf, json, etc.) This information has to be analyzed and the format is crucial to determine whether the tool is more or less effective depending on both, the information to be analyzed and the context.

Data analytics – The ability of exploring data and reports in order to extract meaningful insights in the form of charts and graphs, which can be used to improve the understanding of gathered data.

6. Tool Evaluation Methodology

Previous chapters highlighted the importance of software's evaluation, provided some basic definitions, described the different software categories and evaluation scenarios. This chapter will focus on explaining the proposed software evaluation method and suggest a list of metrics that can be directly used or that can be considered a solid starting point to create new ones.

6.1. Evaluation method overview and requirements

The proposed software evaluation method is based on these key elements:

- Evaluated software belongs to one of the four tool-families (for more on tool-families please consult chapter 1).
- Metrics are used to measure certain software's qualities that are deemed important.
- Similar metrics are collected in macro-groups. Every metrics is weighted and the macro-groups total score for a software depends on the weighted score of all its metrics.
- There are five macro-groups, one is called “general”, the other four “technical”. There is one technical macro-group for each tool-family, it contains specific metrics for its tool-family. The general macro-groups contain universal metrics that can be used for every software.
- Every software is evaluated according to metrics that belong to only 2 of the 5 macro-groups: the general macro-group and the technical for its tool-family.
- Every metrics is evaluated with a score that all into a limited set of numbers that can be negative or positive. Score is assigned according to the principle that “a higher score is better than a lower one”.
- The sum of weighted metrics scores provides a total macro-group score. The sum of weighted macro-group scores provides a total software score.

6.2. Metrics definition & complete list

Metrics, with the meaning of standard of measurement, are the core of the proposed evaluation system. Each metrics has been chosen to reflect one or more desired quality for the software. The proposed metrics are just a guideline; they can be added or removed according to the need of those evaluating the software.

6.2.1. Weighted metrics & weighted macro-groups

Every metrics and every macro-group is weighted. This is useful to convey the idea that some qualities are deemed more important than others.

Metrics' weight is assigned according to these rules:

- All metrics are weighted.
- Each metric's weight is expressed as a percentage.
- Each metric's weight needs to be higher than 0% and smaller than 100%
- The sum of all the weights of metrics that belong to a macro-group must be 100%

Macro-groups are weighted according to these rules:

- All the macro-groups are weighted
- Each macro-group's weight is expressed as a percentage
- Each macro-group's weight needs to be higher than 0% and smaller than 100%
- The sum of all the weights of macro-groups assigned to a software must be 100%

The proposed macro-groups are five, one called “general” and four “technical” assigned to the four tool-families. Every macro-group has the same weight, 50%, this because every tool must be evaluated according to metrics that belong to only two macro-groups (general and one technical) and the combined macro-groups weight must be 100%.

for example: tool X belongs to the tool-family “Information gathering analysis services”, it will be evaluated according to all the metrics contained in the macro-groups “General” and “Technical-IGAS”.

The following table lists all the macro-groups with the number of metrics they contain.

Table 3 – Macro-group overview

Macro-group name	Number of metrics	Notes
General	8	Contains general purpose metrics
Technical – IGAS	5	Contains metrics designed for the tool-family “Information gathering analysis services (IGAS)”
Technical – TAHP	5	Contains metrics designed for the tool-family “Tools for the attack and hook preparation (TAHP)”

Technical – TEAT	5	Contains metrics designed for the tool-family “Tools for the Execution of the attack (TEAT)”
Technical – TIAR	5	Contains metrics designed for the tool-family “Tools for the Information Aggregation and Reporting (TIAR)”

6.2.2. Defining the general macro-group

The general list of metrics originally proposed by the Software Sustainability Institute (see 1.5) can be combined together and reduced to a smaller set of metrics that can still be useful as general purpose metrics to evaluate general qualities of a software. The following table provides the full list of metrics with definition and weights.

Table 4 – Macro-group “general” metrics

Macro-group General		
Metric Name	Weight	Definition
Understandability	20%	How easy is it to understand and learn how to use the software and its functions?
Documentation	15%	Is user documentation comprehensive, appropriate, and well-structured?
Installability	10%	How straightforward is it to build and/or install on a supported system?
Identity	5%	Is Project/software identity clear and unique? Is it easy to understand who owns the project/software?
Support	10%	How easy is to understand how the project is run and the development of the software managed? Is there evidence of current/future community and developer support? Is there any evidence of current/future development?
Portability	5%	Is the software usable on multiple platforms?
Changeability	15%	How easy is it to understand and test at the source level? Is it easy to modify?
Interoperability	20%	Is it interoperable with other required/related software?

6.2.3. Defining the technical macro-groups

All the technical macro-groups contain metrics; these were chosen according to the result of the previously performed analysis of the possible difficulties that must be faced while evaluating tools that belong to each DOGANA tool-family. Please consult chapters 2 to 5 for more information about what are the current scenarios in regards of each tool-family. For each macro-group there is a table that reports that full list of metrics with the corresponding definition and suggested weight.

Table 5 – Macro-group “IGAS” metrics

Macro-group Technique – IGAS – Information gathering analysis services		
Metric Name	Weight	Definition
Number of sources	5%	Number of information sources like social media, documents, public web sites, blogs that the tool is capable to search for.
Performance	15%	A measure of software performance including minimum specific system requirements (the less the better) and time spent for information retrieval, processing and output (the less the better).
correlation capability	20%	Is there any information correlation functionality? If the answer is yes, how many of them and what is the relevance of the gathered information?
output quality	40%	How relevant is the retrieved information with provided search criteria?
Information filtering	20%	Is there any information filtering functionality?

Table 6 – Macro-group “TAHP” metrics

Macro-group Technique – TAHP – Tools for the attack and hook preparation		
Metric Name	Weight	Definition
automation	20%	What is the level of automation in its functions? For example in identifying potential targets, bypassing security challenges, interacting with a “chat environment”.
templating	25%	When it comes to create fake identities, fake profiles or custom made fake web pages, what is the available level of customization? Is it possible to provide different templates or is there only a limited set of premade resources?

impact	20%	Are the most famous social networks and communities included among the exploitable ones? Are there premade versions of famous web sites and/or logos?
Level of variety of the target	10%	How many target social networks, communities and web sites can be targeted/exploited? Do the targets belong to just one category (e.g., only social networks, only chats, etc.) or multiple ones?
Properties of the fake entity that has been created	25%	How good is the tool in emulating human behavior (e.g., chat skills, fake profile creation, etc.) or web pages (e.g., cloning a web site, writing fake emails, etc.).

Table 7 – Macro-group “TEAT” metrics

Macro-group Technique – TEAT – Tools for the Execution of the attack		
Metric Name	Weight	Definition
Multi-attack availability and combination	30%	What is the range of attack vectors and strategies offered by the tool? Is it possible to combine different kind of attacks together? Is it possible to create sequences of attacks?
Automation	30%	Is it possible to automatize the attacking process, either as a whole or single steps of it?
Mass attack level	5%	Is there any functionality regarding the handling of mass attack campaigns? If the answer is “yes”, how many different targets can be attacked in an hour time?
Attacker's identity concealment and or spoofing	30%	Is it possible to hide attacker's identity or assume a fake one? How good are the spoofing/hiding capabilities of the software?
Persistence	5%	Is the tool able to provide some form of persistent access to the target after a successful attack execution?

Table 8 – Macro-group “TIAR” metrics

Macro-group Technique – TIAR - Tools for the Information Aggregation and Reporting		
Metric Name	Weight	Definition
Information structure	20%	Is the post-aggregation report structured in some way? Is data grouped in some nested way with top level data, second level data and so on?
Adaptability/Flexibility	20%	Is the tool usable with different programming languages and/or has any bindings in scripting languages?

Efficiency	20%	A measure of how fast is the tool, how thorough is the examination and how understandable are the results.
Reporting format	20%	Number of exporting formats available and ability in deliver them fast and without technical problems.
Data analytic	20%	The ability of exploring data and reports in order to extract meaningful insights in the form of charts and graphs.

6.3. Score system definition

The key to make the evaluation less subjective and faster is to assign to each metric a raw score value during the evaluation the phase. A well-defined and understood set of discrete values makes evaluation easier to perform and to check.

Only five values are used, they range from 1.0 to -1.0 with increments of 0.5. The proposed score system is based on the assumption that “higher is better”: negative values are useful to apply a “penalty” and positive values denote not just the meeting of the criteria but a very good “performance”, finally a value of 0 is assigned for a normal level.

The values are based on the description found in the following table:

Table 9 – Score system possible values with definition

Score	Definition
+1,0	Fully satisfies the basic requirements and needs only a minimal effort to reach an optimal level. performance.
+0,5	Fully satisfies the basic requirements but needs a minimal amount of effort to reach an optimal level.
0	Satisfies the basic requirements but needs some modification to reach an optimal level.
-0,5	Partially dissatisfies the basic requirements.
-1,0	Fully dissatisfies the basic requirements.

It's worth considering that from the point of view of definitions found in the previous table, there is big difference between two neighbor scores. This feature of the score system is helpful in reducing the complexity of the process; if the evaluation is performed by two people, this

score system increase the chance the both of them are going to assign the same score to the same tool. In order to further reduce the complexity, it is strongly suggested to share a common definition of what is the “basic requirement” for certain metrics.

Please note that a score of 0 is the “bare minimum” requirement, negative ones are assigned when the software has some flaw and positive ones in case of very good software. With this score system the minimum possible score is -100, the maximum is +100 while a score of 0 is assigned to a “basic software”.

Note: Score System, as every part of the evaluation method, should be considered a guideline. Feel free to change the values as desired.

6.4. Performing the evaluation

It's very important to remember that the evaluation method proposed on this chapter is a general guideline. The purpose is to make evaluation as ease and flexible as possible. Metrics, in particular, should be considered guidelines, both in their number as in the weights assigned to them.

6.4.1. Steps to be performed prior to the evaluation phase

Before the evaluation begins it's very important to check the list of tools that need to be evaluated, the list of metrics with their weights and the family tools. Some factors should be considered:

- Is it clear what is the tool-family every software belongs too?
- Is it really necessary to evaluate all the software or some of them automatically deserve to be included or excluded?
- Is there some metrics that can be excluded from the list? Are the weights well balanced?

Changes to the metrics, weights or any other element should be well documented for future reference.

This is a list of task that should be performed before the evaluation phase:

- Every software is assigned to one of the four tool-families.
- A complete list of weighted metrics is compiled, each of them belongs to a macro-group.
- A check is performed to be sure that the sum of all the weights of the metrics of every macro-group is equal to 100%.

- A check is performed to be sure that the sum of the weights of the general macro-group and technical macro-group equals 100%.

6.4.2. Steps to be performed during the evaluation phase

The evaluation process is straightforward and can be broken down in a limited number of steps described in this list:

- A software is selected for evaluation. Its tool-family is known and the corresponding full list of metrics is known too.
- Every metric is evaluated according to the score system.
- Once all metrics have been received the total score for the macro-groups is calculated.
- The final software's score is calculated from the macro-groups scores.

The evaluation phase is a critical moment; it can highlight some issue with the current evaluation choices. Maybe the number of metrics is too large or the weights are not well balanced. In any case this can be the right moment to stop and change some parameter. In this case it's good to interrupt the evaluation and consider it part of the previous step (see 6.4.1). Some changes can be made to the evaluation rules and a new evaluation can start again, hopefully without any further problems.

6.4.3. Steps to be performed after the evaluation phase

Once all the software tools have been evaluated, it's possible to compile a list of tools divided by tool-family. Sometimes the scores are enough to decide which tool include and which not, sometimes the numbers are not that helpful and other kind of considerations can be helpful. We should keep in mind that whatever the final evaluation methodology is going to be, it's very important to state clearly what is going to be the role of the final score in the decision making phase (i.e., is the score the only parameter taken into account for the final decision? or are there others like, for example, the personal experience of those involved in the evaluation process?).

6.5. An example of report for the evaluation of two tools

This paragraph is a step by step example of how to perform an evaluation with the method described in this chapter according to the steps described in 6.4.2.

Step 1

Two tools need to be evaluated, in this example they are called software1 and software2. They belong to the tool-family “Tools for the Information Aggregation and Reporting”.

They are going to be evaluated according to 13 metrics, divided in 2 groups, as shown in this table:

Table 10 – Evaluation example – complete list of metrics

Macro-group General	Macro-group Technical-TIAR
Understandability, Documentation, Installability, Identity, Support, Portability, Changeability, Interoperability	Information structure, Adaptability/Flexibility, Efficiency, Reporting Format, Data Analytic.

Step 2 & Step 3

Every metric is evaluated according to the score system, and the weight is applied to calculate the weighted score. For each macro-group the sum of the scores of all the metrics is calculated to achieve the total score for the macro-group. The result can be shown in a table for each macro-group.

Table 11 – Evaluation example – macro-group “general” scores

		SOFTWARE ALTERNATIVES			
		software 1		software 2	
	Weight	Raw	Weighted	Raw	Weighted
Macro-group GENERAL	50%				
Understandability	20%	1.0	20%	0	0%
Documentation	15%	0.5	7.5%	0	0%
Installability	10%	0.5	5%	1.0	10%
Identity	5%	1.0	5%	0	0%
Support	10%	0	0%	1.0	10%
Portability	5%	-1.0	-5%	0	0%
Changeability	15%	0	0%	1.0	15%
Interoperability	20%	0	0%	1.0	20%
Subtotal			32.5%		55%

Table 12 – Evaluation example – macro-group “TIAR” scores

		SOFTWARE ALTERNATIVES			
		software 1		software 2	
	Weight	Raw	Weighted	Raw	Weighted
Macro-group Technical-TIAR	50%				
Information structure	20%	0.5	10%	0.5	10%
Adaptability/Flexibility	20%	1.0	20%	1.0	20%
Efficiency	20%	0.5	10%	0	0%
Reporting format	20%	1.0	20%	1.0	20%
Data analytic	20%	0.5	10%	1.0	20%
Subtotal			70%		70%

Step 4

The final software's score is calculated from the macro-groups scores that need to be weighted.

Table 13 – Evaluation example – final scores

Macro-group		Software alternatives			
		Software1 name		Software2 name	
Name	Weight	Raw	Weighted	Raw	Weighted
General	50%	32.5%	16.25%	55%	27.5%
Technical	50%	70%	35%	70%	35%
Total	100%	51.25%		62.5%	

7. Software Licenses

DOGANA framework will contain several software components but details about how these will be integrated together are not yet defined. Chosen components' software licenses will have a high impact on the final framework license type and as a logical consequence also on the software evaluation phase.

This chapter's aim is to provide an overview of the available open source licenses, explain the differences between them and rating them according to a parameter called “openness” (i.e., a number in the range of 1 to 4, where 1 means “the least open license” and 4 indicates “the most open”).

Table 14 – Licenses' overview

	Required	Permitted	Forbidden	Openness
Public domain Dedication (The Unlicense)		Commercial Use Distribution Modification Private Use	Hold liable	4
MIT	License and copyright notice	Commercial Use Distribution Modification Patent Use Private Use	Hold liable	3
BSD (ISC License)	License and copyright notice	Commercial Use Distribution Modification Private Use	Hold liable	3
Apache 2.0	State change License and copyright notice	Commercial Use Distribution Modification Patent Use Private Use	Hold liable Use trademark	3
Artistic license 2.0	License and copyright notice State Changes	Commercial Use Distribution Modification Patent Use Private Use	Hold liable Use trademark	3

GNU LGPL	State change License and copyright notice Same License	Commercial Use Distribution Modification Patent Use Private Use	Hold liable	2
Mozilla Public License 2.0	Disclose Source License and copyright notice Same License	Commercial Use Distribution Modification Patent Use Private Use	Hold liable Use trademark	2
Eclipse Public License 1.0	Disclose Source License and copyright notice Same License	Commercial Use Distribution Modification Patent Use Private Use	Hold liable	2
GNU GPL 3	Disclose Source License and copyright notice Same License State Changes	Commercial Use Distribution Modification Patent Use Private Use	Hold liable	1

The previous table makes us of some technical definitions that are reported here for the sake of clarity. Definitions are divided according to the column where they can be found (e.g., forbidden, permitted and required); please notice that some definition can be found, with the same name, in different columns with a different meaning.

Column “Forbidden”:

- **Hold Liable** – Software is provided without warranty and the software author/license owner cannot be held liable for damages.
- **Patent Use** – This license explicitly states that it does NOT grant you any rights in the patents of contributors.
- **Trademark Use** – While this may be implicitly true of all licenses, this license explicitly states that it does NOT grant you any rights in the trademark or other marks of contributors.

Column “Permitted”:

- **Commercial Use** – This software and derivatives may be used for commercial purposes.
- **Distribution** – You may distribute this software.
- **Modification** – This software may be modified.
- **Patent Use** – This license provides an express grant of patent rights from the contributor to the recipient.
- **Private Use** – You may use and modify the software without distributing it.

Column “Required”:

- **Disclosure Source** – Source code must be made available when distributing the software.
- **License and Copyright Notice** – Include a copy of the license and copyright notice with the code.
- **Network Use is Distribution** – Users who interact with the software via network are given the right to receive a copy of the corresponding source code.
- **Same License** – Modifications must be released under the same license when distributing the software. In some cases, a similar or related license may be used.
- **State Changes** – Indicate significant changes made to the code.

Table 15 – Notes on the licenses.

	Notes
Public domain Dedication (The Unlicense)	There are two versions of the Public Domain Dedication license, we're referring to the one named “The Unlicense”. Since copyright is automatic in most countries, this license type is basically just a template to give up any kind of copyright interest. The software is dedicated then to the public domain. This license should be used to opt out of copyright entirely.
MIT	A short license that allows a great degree of freedom in regards of the use of software. As long as there is proper attribution and the lack of warranty is accepted, people can basically do anything with the code.
BSD (USC License)	The BSD license considered here is called ISC License and is functionally equivalent to the MIT license. People can do anything with the code as long as there is proper attribution and the lack of warranty is accepted.
Apache 2.0	A license that provides express grant of patent rights from contributors to users.

Artistic license 2.0	A license that requires that modified versions of the software do not prevent users from running the standard version.
GNU LGPL 3	GNU LGPLv3 is a license that requires that derived works be licensed under the same license. this restriction doesn't apply to works that only link to it.
Mozilla Public License 2.0	This license is currently maintained by the Mozilla foundation and tries to be a good compromise between two licenses: the reciprocal GPL and the more permissive BSD license.
Eclipse Public License 1.0	This license has three interesting features: provides the ability to commercially license binaries, linked works can use other licenses including commercial ones, provides a modern royalty-free patent license grant. Eclipse Public License is basically a commercially-friendly copyleft license.
GNU GPL 3	The most widely used free software license. The copyleft requirements are quite strong: source code of the work must be made available under the same license when distributing derived works.

Four types of licenses can be distinguished here:

- **Public domain:** Software placed in the public domain (no copyright or ownership). Components under this license can be used, copied, modified, distributed or sub-licensed. This is the **most favorable case**.
- **Permissive or non-protective free open-source software license (permissive FOSS):** Components under this license can be used, copied, modified, distributed and sub-licensed (under attribution). **Most open-source favorable type of license.**
 - MIT, BSD, APACHE licences, Artistic license
- **Weakly protective free open-source software license (weakly protective FOSS):** Components under this license can be used, copied and modified. Distribution has to be under the same license only when the code has been modified. When no modification has been made, sub-licensing is possible.
 - LGPL, Mozilla Public, Eclipse licenses
- **Protective free open-source software license (protective FOSS):** Components under this license can be used, copied, and modified. Distribution has to be in the same license and no rights to sub-license

GNU GPL

8. Awareness Solutions

Establishing a minimum awareness level for all personnel can be the base of the security awareness program. Security awareness may be delivered in many ways, including formal training, computer-based training, e-mails and circulars, memos, notices, bulletins, posters, etc.

The training program should require personnel to acknowledge they have received and understand the content being delivered. This is crucial to the success of the security awareness program. If content is being delivered and not understood, the employee may still inadvertently put the organization's information at risk and vanish the investment at the same time. Feedback on training content and comprehension are key to ensuring personnel understand the content and the organization's security policies. [13]

Metrics can be an effective tool to measure the success of a security awareness program, and also provide valuable information to keep the security awareness program up-to-date and effective. The particular metrics used to measure the success of a security awareness program will vary for each organization based on considerations such as size, industry, and type of training.

Awareness metrics differs from the other evaluation metrics of this document. A part of the performances of an awareness program can only be measured using the Social Driven Vulnerability Assessment (SDVA) toolchain, which is an outcome of the DOGANA project. SDVA, performed through DOGANA framework, are actually measuring a risk, which an appropriate awareness program aims to reduce.

The effectiveness of the awareness programs are hence measured in a negative retrofitting loop, using again the SDVA and detecting that the original risk has lowered (see Figure 1). Summing up, the DOGANA framework can be the way through which the performances of the awareness programs are measured, for example applying it before and after an awareness initiative, at scheduled intervals. However, the awareness metrics are not used as a metric for the design of the DOGANA framework, but rather as an indicator of the performances of the countermeasures.

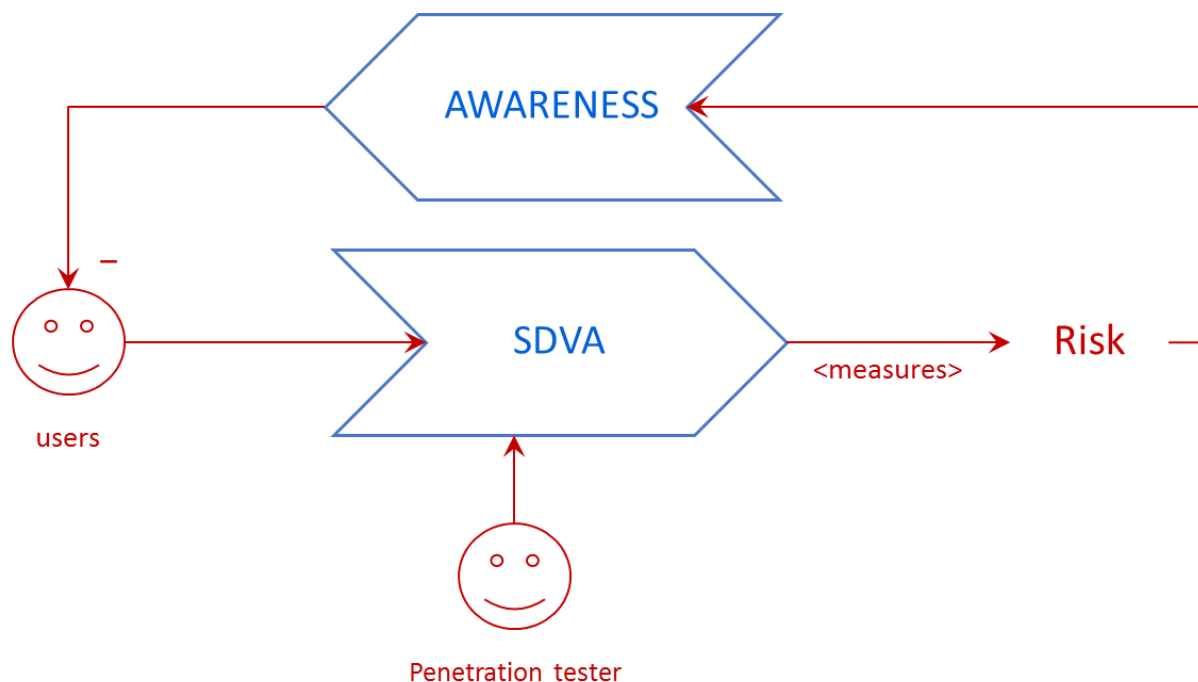


Figure 1 – Simplified view of the role of SDVA as a retrofitting tool, used to improve performance of awareness programs and ultimately to reduce risk

8.1. Conceptual model

Simplifying, the ultimate result of any awareness programs is to increment the overall robustness of the information space where an asset (the one the attackers wants to steal) resides. As described in Chapter 1 of Deliverable D2.1 the information space is composed by two elements: the human and the technological resources, both handling an asset as a chain of business processes. We propose to still use the schema of D2.1, see Figure 2, as a model to back the selection of metrics for the awareness programs with a coherent view.

With reference to Figure 2 the information space could be seen as a waterfall process (a concatenation of two business processes), where the user (who plays the role of the awareness program recipient) has an impact on the technical information systems of the enterprise (e.g., less incidents due to increased ability to discriminate phishing).

The awareness is an input to the system of Figure 2, which has two consequences:

- The employee better discriminates SE-twisted attacks. He is the recipient of an awareness program, who trains himself using different methods, more or less effective. Whatever method it is used the effect of an awareness program is to improve the resilience of the employee, leveraging his ability to discriminate SE twisted attacks (e.g., spear phishing mails). This increased resilience impacts on the overall information space robustness. This is a **first level** consequence of the awareness program. These

metrics are measurable from the humans/employees for example through questionnaires.

- The enterprise systems intervene less times. The solutions that the enterprise uses to prevent SE-twisted attacks (e.g., threat intelligence systems, see deliverable D2.1, Chapter 6) intervene less times if the employees are able to discriminate SE-twisted attacks. This is a **second level** consequence of the awareness program. These metrics are more or less directly measurable from the security information system.

Each level has its own set of metrics and measures of both are mandatory to assess the effectiveness of an awareness program. In other words, the effectiveness of an awareness program can be done at the first level, measuring the attitudinal changes of the users or at the second level, measuring the indirect effects that these changes had on the system (e.g., less warnings raised when a scam sites is visited from within the enterprise).

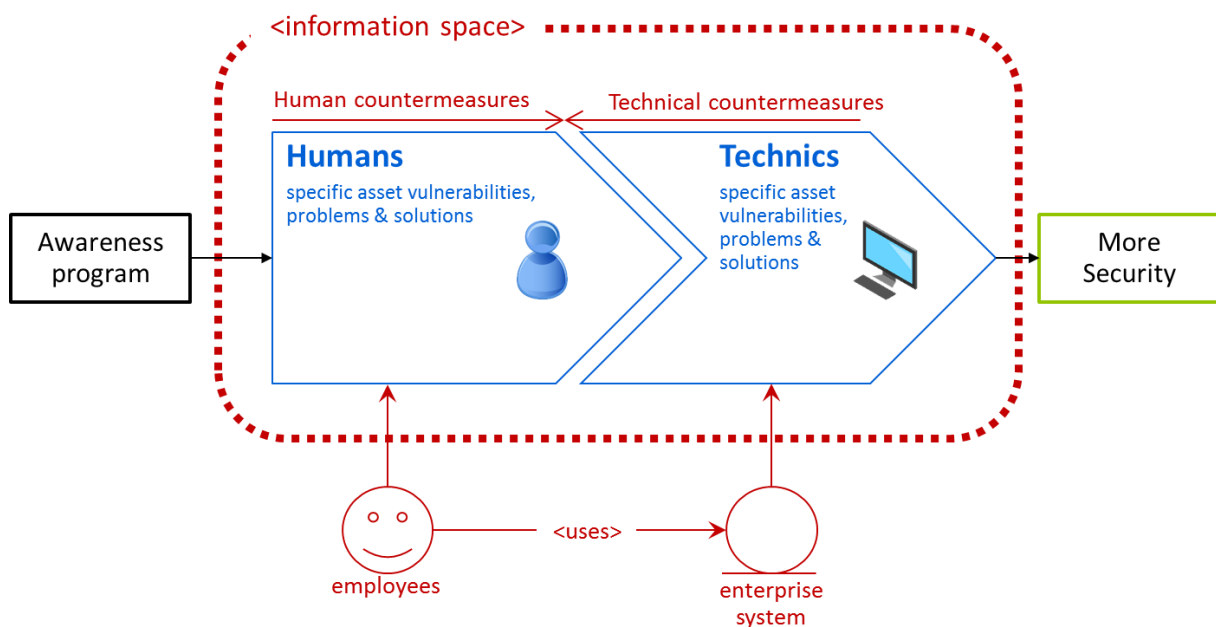


Figure 2 – A model of the information space as a business process where the user, attending the awareness program, has an impact on the information systems

The **first level** metrics are defined around a model of humans (actually employees) which cannot be over simplified. In particular, we propose an initial classification of attributes, starting from the scientific literature, which could be further explored:

- User seen as a “human-being”. It considers the common neural attitudes that are related to factors like race, culture and country of origin [14][15].

- User as a “part of a category within an organization”. It considers the human focusing on the role and competence he plays within the company he works for (e.g., different program for secretary of CSO or developers).
- User as a “single individual”. It considers all the psychological factors and personal history that go to build the character of a single person [16][17].

8.2. List of possible metrics which measure awareness effectivity

Metrics can then be divided into two major categories.

First level effectiveness metrics – these metrics measure **first level** changes, directly impacting the user, according to a user model:

- **Number personnel completing training** measured through attendance tracking and performance evaluations, which indicates the potential diffusion of the awareness delivery in terms of coverage of the company population.
- **Number of employees with privileged access who have received required training**, which is related to particular employees with privileged rights and it is also measured through attendance tracking and performance evaluations.
- **Personnel comprehension of training material**, which indicates how employees understood the topic explained during the training. It could be measured through quiz, interactive games, or other test mechanisms, which can measure the level of knowledge of employees.
- **Employees evaluation of feedback**, which indicates the level of satisfaction of who receive the awareness. This measure, obtained for example through a survey, could allow to understand if employees appreciate the awareness method and indirectly if it could be useful.

Second level effectiveness metrics – these metrics measure the second level effects. These metrics are more easily measurable from the enterprise systems:

- **Number of reports of attempted e-mail or phone scams**. The increase of this metric could indicate a better recognition by personnel of phishing and other social-engineering attempts.
- **Number of times the identity theft protection system intervened**. The systems implemented to prevent accesses from unknown location (e.g., a foreign country due to identity theft) intervene less, for example because the employees do not anymore yield his credentials on scam sites or better handle passwords.
- **Number of queries from personnel on how to implement secure procedures**. The fact that personnel is aware of security procedure can be a positive outcome of

training program. Measuring the requests related to how to implement a secure procedure could indicate the level of awareness of the company.

- **Time to address and mitigate potential attacks.** The reduction of this time could indicate Better understanding by personnel of potential threats and risks to sensitive information
- **Number of infected computers.** It is intended as the reduction in malware outbreaks and computer performance issues related to malware. This metric could be useful because most infected workstations are a results of human behavior. As employees are trained this measure should decrease over time.

8.3. DOGANA metrics

The model of Figure 2 introduced above, requires a metric that measure the overall robustness against SE attacks of the whole information space. This category of metrics usually is the result of SDVA performed for example using the DOGANA framework. For example, DOGANA will simulate a phishing attack against an enterprise and its results can be the ground for evaluating the effectiveness of a previous awareness campaign. SDVA planned at regular intervals could help to understand the real impact of the awareness programs activated. These metrics are not either primary or secondary because they rather measure the overall robustness of the enterprise protection systems, both human and technological. They are hence metrics of the whole information space.

The following list reports some examples of this kind of metrics:

- **Number of users who click on a phishing link** – A phishing assessment has the goal to measure who falls victim to such attacks. This measure can indicate the security posture of the company. This number should decrease over time as positive behavioral change.
- **Number of user who lose a sensitive asset** – A phishing assessment could also include the measure of who lose a sensitive asset. This measure indicates even a worse risk with respect to the previous one. This number should decrease over time as positive behavioral change.
- **Number of phishing detection** – A phishing simulation could be useful also to evaluate user reaction with respect to security procedure: in this sense, measuring the number of people who detect and report a phishing attack could be useful (it is useful to remember that the SDVA could either deliver a drive-by-infection or a drive-by-download threat). This number should increase over time. Moreover, it could be interesting monitoring the actual destination of report, in order to evaluate compliance to enterprise procedures.

- **Sensitive data presence on social media** – Presence of company sensitive data on social media is an indicator of how employees are aware regarding risk related to information sharing (measurement of the enterprise’s digital shadow). Monitoring social media and measuring number of contents regarding organization information publicly available on social network could indicate the level of awareness.
- **Types of information leaked in a phishing test** – SDVA usually test users asking to concede some sensitive information (usually the enterprise credentials) to a scam site. Not all the users insert real or complete credentials. The type of information leaked is an indicator of the level of trust the users have in the scam.

9. Conclusions

This document presented the result of the work done to provide DOGANA framework with a unified methodology to perform evaluation and quality measurement in several scenarios: software evaluation (i.e., custom developed software and third party tools to be integrated), awareness campaign success level evaluation, Technology Readiness Level measurement.

During an initial phase of the work, a part of the consortium identified different evaluation scenarios and focused on defining a general approach to face the “evaluation requirements” of the projects with the correct level of details but with a lightweight solution. The result of this work can be seen mostly in Chapter 1.

The next phase of work focused mostly on the software evaluation aspect of the task, specifically on the four families of tools identified in the first chapter. For each software category, the consortium started by analysing current trends, technologies used (e.g., software languages) and describing in broader terms what kind of software solutions are available (e.g., libraries, web application, stand-alone software). For each tool family, then a list of possible problems associated with evaluating process has been listed. Key elements to be taken into account during an evaluation process has been suggested along with guidelines. The result of this phase of the work is available in Chapters 2 to 5; each one dedicated to a single tool family.

Information provided in these four chapter is far from being complete due to the large amount of available tools and technologies. The result is nonetheless very important for the definition of a lightweight methodology for software evaluation. DOGANA requirements are too specific (i.e., consider privacy issues, legal compliance requirements) to be simply fulfilled by blindly picking the most known software solutions available in the market without any regards for licenses, technological requirements or privacy and legal concerns.

Once different scenarios and relative problems were clear, a third phase started and focused not just on software evaluation but on all different kind of measurement required by DOGANA. In a parallel and independent way, the work continued on three different topics: defining a

software evaluation methodology (i.e., according to what suggested in previous chapters), defining an awareness campaign success evaluation methodology, analysing the current approaches to evaluate the Technological Readiness Level of a project. Meanwhile a research on suitable software licenses for DOGANA has also been performed. The result of the work on a software evaluation methodology is reported in Chapter 6 while Chapter 7 contains the report on different suitable license types for DOGANA, Chapter 8 presents the methodology for awareness campaign related evaluation and finally Annex I reports the findings on Technology Readiness Level metrics.

The results of this document will be useful during in different phases of the project. Some chapter present results that will be used during the software evaluation phase (i.e., see D3.1 for more information) while others will be used at the end of the project or after an awareness campaign. It can be said that, the result of Chapter 1 has been during the development of the rest of the document while the result of Chapters from 2 to 5 has been useful for the development of Chapter 6 but is consider also as important for those actually involved in the software evaluation phase. Chapter 6 and 8 present real methodologies that should be used during evaluation. Chapter 7 presents useful guidelines to decide which software licenses should be considered acceptable or not. Finally, Annex I will be useful for an internal evaluation of DOGANA TRL.

10. Annex I: Technology Readiness Metrics (TRL)

The aim of this annex is to provide a glance at the methods to evaluate the Technology Readiness Level. The annex, which goes beyond the scope of task T2.2 is not intended to have a direct impact on the task which will benefit from this deliverable (T3.1, T6.1), but instead to start paving the way toward the evaluation of the whole DOGANA framework. It starts by listing a set of relevant and publicly available TRL calculation methodologies and thus goes a bit into deep of each specific methodology, also highlighting the aspects of each methodology that may be relevant for DOGANA.

10.1. Introduction

In literature it is possible to find many different proposals to establish the Technology Readiness Level (TRL) of systems and processes. The most interesting for DOGANA are the following:

- The NASA Technology Readiness Level (TRL) [19].
- The Department Of Homeland Security (DHS) Science and Technology (S&T) Readiness Level [20].
- The European Commission (EC) Technology Readiness Level (TRL) definition introduced in the Horizon 2020 programme [21].
- The Technology Readiness Level (TRL) defined by ISO Standard 16290:2013 for space systems [22].
- The U.S. Army Communications Electronics Command (CECOM) Technology Readiness Level (TRL) for both hardware/subsystems and software [23].
- The U.S. Army Software Readiness Level (SRL) [24].

A brief overview of the possible definitions is reported in the following.

10.2. The NASA TRL levels

The Technology Readiness Level (TRL) scale was initially developed during the 1970-80's. The National Aeronautics and Space Administration (NASA) introduced the scale as *"a discipline-independent, program figure of merit (FOM) to allow more effective assessment of, and communication regarding the maturity of new technologies"* (see Table 16).

Table 16 – NASA TRL levels

TRL	Definition
TRL 1	Basic principles observed and reported
TRL 2	Technology concept and/or application formulated
TRL 3	Analytical and experimental critical function and/or characteristic proof-of-concept
TRL 4	Component and/or breadboard validation in laboratory environment
TRL 5	Component and/or breadboard validation in relevant environment
TRL 6	System/subsystem model or prototype demonstration in a relevant environment (ground or space)
TRL 7	System prototype demonstration in a space environment
TRL 8	Actual system completed and - flight qualified through test and demonstration (ground or space)
TRL 9	Actual system - flight proven through successful mission operations

10.3. DHS Science and Technology Readiness Level

The DHS has introduced, starting from the NASA TRL levels, a modification of the TRL scale by including also:

- The Integration Readiness Levels (IRLs) addressing the integration of a component technology into a complete system.
- The System Readiness Levels (SRLs) indicating the level of maturity applied at the system-level.
- The Manufacturing Readiness Levels (MRLs) to evaluate the “manufacturing readiness” of a product.
- The Programmatic Readiness Levels (PRLs) to address program management concerns.

There are interesting aspects of DHS work that may have an impact on DOGANA:

- A modified version of the TRL definition and a clustering of several TRL into 3 main categories (see Table 17).
- The definition of a set of questions to be answered for each readiness level to properly support evidence and completeness when classifying a system or a product in a given readiness category. It shall be considered that not all the questions identified in DHS (reported in Table 18) are relevant to DOGANA, even if

they still represent a solid basis for the development of a set of questions tailored to fit the DOGANA needs.

- The development of a TRL calculator (Figure 3) based on Microsoft Excel that allows the assessment of TRL, PRL and MRL. The TRL calculator uses of a scale from 0% (question not fulfilled at all) to 100% (question successfully answered) to evaluate the level of completeness and a color coded (red, yellow, green) level of completeness. Similarly, to the previous point, it shall be considered that the Microsoft Excel file proposed doesn't directly fit the DOGANA needs, but should be only considered as a possible starting point.

Table 17 – DHS modified TRL

Cluster	TRL
Research and Development	TRL 2
	TRL 3
Testing and Demonstration	TRL 5
	TRL 6
	TRL 7
Product and Deployment	TRL 9


Table 18 – DHS TRL questions

TRL	Definition
TRL 1	Do rough calculations support the concept?
	Do basic principles (physical, chemical, mathematical) support the concept?
	Does it appear the concept can be supported by software?
	Are the software requirements known in general terms?
	Do paper studies confirm basic scientific principles of new technology?
	Have mathematical formulations of concepts been developed?
	Have the basic principles of a possible algorithm been formulated?
	Has a scientific methodology or approach been developed?
TRL 2	Has potential system or component applications been identified?
	Have paper studies confirmed system or component application feasibility?
	Has an apparent design solution been identified?
	Have the basic components of the technology been identified?


TRL	Definition
	<p>Has the user interface been defined?</p> <p>Have technology or system components been at least partially characterized?</p> <p>Have performance predictions been documented for each component?</p> <p>Has preliminary software coding that confirms basic principles been documented?</p> <p>Has a functional requirements generation process been initiated?</p> <p>Does preliminary analysis confirm basic scientific principles?</p> <p>Have experiments validating the concept been performed with synthetic data?</p> <p>Are basic scientific principles confirmed with analytical studies?</p> <p>Do all individual parts of the technology work separately? (No real attempt at integration)</p> <p>Is the hardware that the software will be hosted on available?</p> <p>Are output devices available?</p>
TRL 3	<p>Have predictions of components of technology capability been validated?</p> <p>Have analytical studies verified performance predictions and produced algorithms?</p> <p>Can all science applicable to the technology be modeled or simulated?</p> <p>Is outline of software algorithms documented?</p> <p>Do experiments or modeling and simulation (M&S) validate performance predictions of technology capability?</p> <p>Does preliminary coding verify that software can satisfy an operational requirement?</p> <p>Do experiments verify feasibility of application of technology?</p> <p>Do experiments or modeling and simulation (M&S) validate performance predictions of components of technology capability?</p> <p>Have cross-technology effects (if any) been identified?</p> <p>Do paper studies indicate that technology or system components can be integrated?</p> <p>Are the technology or system performance metrics established?</p> <p>Have technology or system performance characteristics been confirmed with representative data sets?</p> <p>Do algorithms run successfully in a laboratory environment, possibly on a surrogate processor?</p> <p>Has inventory of available software that does similar tasks been completed?</p> <p>Has existing software been examined for possible reuse?</p> <p>Has scientific feasibility of proposed technology been fully demonstrated?</p> <p>Does analysis of present technologies show that proposed technology or system fills a capability gap?</p>
TRL 4	<p>Have cross-technology effects (if any) been fully identified and documented?</p> <p>Has acceptance testing of individual components been performed?</p> <p>Has performance of components and interfaces between components been demonstrated?</p> <p>Does draft system architecture plan exist?</p>

TRL	Definition
	<p>Have end user technology/system requirements been documented?</p> <p>Does breadboard demonstrate functionality of all components?</p> <p>Have algorithms been converted to pseudocode?</p> <p>Has analysis of data requirements and formats been completed?</p> <p>Do stand-alone modules align with preliminary system architecture plan?</p> <p>Has component compatibility been demonstrated?</p> <p>Does technology demonstrate basic functionality in simplified environment?</p> <p>Have performance characteristics been demonstrated in a laboratory environment?</p> <p>Does prototype solve synthetic full-scale problems or process fully representative data sets?</p> <p>Have all functions or modules been demonstrated in a laboratory environment?</p> <p>Have ad hoc integration of functions or modules been demonstrated?</p> <p>Have low-fidelity assessments of system integration and engineering been completed?</p>
TRL 5	<p>Have cross-technology effects (if any) been fully identified, analyzed, and documented?</p> <p>Have internal system interface requirements been documented?</p> <p>Have external interfaces been documented?</p> <p>Has analysis of internal interface requirements been completed?</p> <p>Does the breadboard have realistic interfaces?</p> <p>Is coding of individual functions/modules completed?</p> <p>Can all system specifications be simulated and validated within a laboratory environment?</p> <p>Has a brassboard been developed?</p> <p>Is the laboratory environment high-fidelity?</p> <p>Technology Readiness Level Questions</p> <p>Have functions been integrated into modules?</p> <p>Have individual component functions been verified through testing?</p> <p>Have system modules been debugged?</p> <p>Has integration of modules/functions been demonstrated in a laboratory environment?</p> <p>Have algorithms been run on a processor that can be fielded in an operational environment?</p> <p>Have objective and threshold operational requirements been developed?</p> <p>Has a Product Breakdown Structure been developed?</p>
TRL 6	<p>Have system integration issues been addressed?</p> <p>Is the operational environment fully known?</p> <p>Have performance characteristics been verified in a simulated operational environment?</p> <p>Has prototype been tested in a simulated operational environment?</p> <p>Has system been tested in realistic environment outside the laboratory?</p>

TRL	Definition
	<p>Has an inventory of external interfaces been completed?</p> <p>Has analysis of timing constraints been completed with satisfactory results?</p> <p>Has analysis of database structures and interfaces been completed?</p> <p>Does the prototype functionally handle realistic problems?</p> <p>Have software algorithms been integrated with existing systems?</p> <p>Has functionality of integrated modules been tested?</p> <p>Is software documentation available?</p> <p>Has engineering feasibility been fully demonstrated?</p>
TRL 7	<p>Can unavailable system components be simulated using modeling and simulation (M&S)?</p> <p>Have all interfaces been tested individually under stressed and anomalous conditions?</p> <p>Do algorithms run on processor(s) in an operational environment?</p> <p>Has technology or system been tested in a relevant environment?</p> <p>Are available components representative of production components?</p> <p>Has operational testing of technology/system in relevant environment been completed?</p> <p>Has fully integrated prototype been demonstrated in actual or simulated operational environment?</p>
TRL 8	<p>Are all technology/system components form, fit, and function compatible?</p> <p>Is technology/system form, fit, and function compatible with operational environment?</p> <p>Has technology/system form, fit, and function been demonstrated in operational environment?</p> <p>Has software been thoroughly debugged?</p> <p>Is technical Developmental Test and Evaluation (DT&E) successfully completed?</p>
TRL 9	<p>Does technology/system function as defined in Operational Concept document?</p> <p>Has technology/system has been deployed in intended operational environment?</p> <p>Has technology/system been fully demonstrated?</p> <p>Technology Readiness Level Questions</p> <p>Has Operational Test and Evaluation (OT&E) been successfully completed?</p>



DHS S&T Readiness Level Calculator (ver 1.1)



Project Name:

Project Manager:

Date Computed:

[Glossary](#)

INSTRUCTIONS:

1. Enter program identification information above.
2. Select the types of questions you wish to include on the RL Calculator worksheet. **You do not have to choose a level from the resulting drop down menu. Default settings will start the RL Calculator questions at level 1.** However, if you wish to start at a given level select it from list.
3. Click the "Continue to Calculator" button.

☒

Hardware

☐

Software

☒

Both Hardware and Software

☒ Use

Technology Readiness Level

☐ Omit

If you would like to start at a given level, based on the definitions below, select that level here.

TRL 9	<input type="radio"/>	Actual system proven through successful mission operations	Full definition
TRL 8	<input type="radio"/>	Actual system completed and qualified through test and demonstration	Full definition
TRL 7	<input type="radio"/>	System prototype demonstration in an operational environment	Full definition
TRL 6	<input type="radio"/>	System/subsystem model or prototype demonstration in a relevant environment	Full definition
TRL 5	<input type="radio"/>	Component and/or breadboard validation in relevant environment	Full definition
TRL 4	<input type="radio"/>	Component and/or breadboard validation in laboratory environment	Full definition
TRL 3	<input type="radio"/>	Analytical and experimental critical functions and/or characteristic proof-of-concept	Full definition
TRL 2	<input type="radio"/>	Technology concept and/or application formulated	Full definition
TRL 1	<input type="radio"/>	Basic principles observed and reported	Full definition
	<input type="radio"/>	None of the above	

☐ Use

Manufacturing Readiness Level

☐ Omit

If you would like to start at a given level, based on the definitions below, select that level here.

MRL 9	<input type="radio"/>	Manufacturing processes proven	Full definition
MRL 8	<input type="radio"/>	Manufacturing process maturity demonstration	Full definition
MRL 7	<input type="radio"/>	Prototype manufacturing system	Full definition
MRL 6	<input type="radio"/>	Critical manufacturing processes prototyped	Full definition
MRL 5	<input type="radio"/>	Manufacturing process development	Full definition
MRL 4	<input type="radio"/>	Laboratory manufacturing process demonstration	Full definition
MRL 3	<input type="radio"/>	Manufacturing concepts identified	Full definition

☐ Use

Programmatic Readiness Level

☐ Omit

If you would like to start at a given level, based on the definitions below, select that level here.

PRL 9	<input type="radio"/>	Safety and Training is complete	Full definition
PRL 8	<input type="radio"/>	Training and Test and Evaluation Documentation are complete	Full definition
PRL 7	<input type="radio"/>	Finalized Verification, Validation and Accreditation of system	Full definition
PRL 6	<input type="radio"/>	Formal requirement documents, TEMP, and Systems Engineering Plan complete	Full definition
PRL 5	<input type="radio"/>	Systems engineering and architecture and end user involvement are established	Full definition
PRL 4	<input type="radio"/>	IPTs and working groups for developing and transitioning technology are established	Full definition
PRL 3	<input type="radio"/>	Program risk, requirements, and performance characteristics and measures are determined	Full definition
PRL 2	<input type="radio"/>	Establishment of program with identified customer and technology	Full definition
PRL 1	<input type="radio"/>	Identification of basic scientific concepts and performers	Full definition

Continue to Calculator

Figure 3 – The DHS TRL calculator interface

10.4. European Commission Technology Readiness Level

The EC has defined, for its last research and development programme Horizon 2020 (H2020) 9 different Technology Readiness Levels as reported in Table 19.

No further tools or documents for implementing the metric are provided.

Table 19 – Technology Readiness Level (TRL)

TRL	Definition
TRL 1	Basic principles observed
TRL 2	Technology concept formulated
TRL 3	Experimental proof of concept
TRL 4	Technology validated in lab
TRL 5	Technology validated in relevant environment (i.e., industrially relevant environment in the case of key enabling technologies)
TRL 6	Technology demonstrated in relevant environment (i.e., industrially relevant environment in the case of key enabling technologies)
TRL 7	System prototype demonstration in operational environment
TRL 8	System complete and qualified
TRL 9	Actual system proven in operational environment (i.e., competitive manufacturing in the case of key enabling technologies; or in space)

10.5. ISO 16920 Technology Readiness Level

ISO 16290:2013 defines Technology Readiness Levels (TRLs). It is applicable primarily to space system hardware, although the definitions could be used in a wider domain in many cases.

The definition of the TRLs provides the conditions to be met at each level, enabling accurate TRL assessment. However, conditions are too specific for space systems to be applied to Social Engineering tools.

Table 20 – ISO 16290 Technology Readiness Level (TRL)

TRL	Definition
TRL 1	Basic principles observed and reported
TRL 2	Technology concept and/or application formulated
TRL 3	Analytical and experimental critical function and/or characteristic proof-of-concept
TRL 4	Component and/or breadboard functional verification in laboratory environment
TRL 5	Component and/or breadboard critical function verification in relevant environment
TRL 6	Model demonstrating the critical functions of the element in a relevant environment
TRL 7	Model demonstrating the element performance for the operational environment
TRL 8	Actual system completed and accepted for flight ("flight qualified")
TRL 9	Actual system "flight proven" through successful mission operations

10.6. CECOM Technology Readiness Level (TRL)

The objective of the CECOM study was to assess the feasibility of developing an information assurance (IA) technology readiness level (TRL) assessment method (or equivalent) for technologies at various TRLs to reduce the risk associated with investing in immature technologies.

For each TRL, descriptions are given for hardware/subsystems (HW/S), and software (SW) as shown in Table 21.

Table 21 – Technology Readiness Level (TRL)

TRL	Definition	Description
TRL 1	Basic principles observed and reported	<p>HW/S: Lowest level of technology readiness. Scientific research begins to be translated into applied research and development. Examples might include paper studies of a technology’s basic properties.</p> <p>SW: Lowest level of software readiness. Basic research begins to be translated into applied research and development. Examples might include a concept that can be implemented in software or analytic studies of an algorithm’s basic properties.</p>
TRL 2	Technology concept and/or application formulated	<p>HW/S and SW: Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.</p>
TRL 3	Analytical and experimental critical function and/or characteristic proof of concept	<p>HW/S: Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.</p> <p>SW: Active research and development is initiated. This includes analytical studies to produce code that validates analytical predictions of separate software elements of the technology. Examples include software components that are not yet integrated or representative but satisfy an operational need. Algorithms run on a surrogate processor in a laboratory environment</p>
TRL 4	Component and/or breadboard validation in laboratory environment	<p>HW/S: Basic technological components are integrated to establish that they will work together. This is relatively “low fidelity” compared to the eventual system. Examples include integration of ad hoc hardware in the laboratory.</p> <p>SW: Basic software components are integrated to establish that they will work together. They are relatively primitive with regard to efficiency and reliability compared to the eventual system. System software architecture development initiated to include interoperability, reliability, maintainability, extensibility, scalability, and security issues. Software integrated with simulated current/legacy elements as appropriate.</p>

TRL	Definition	Description
TRL 5	Component and/or breadboard validation in relevant environment	<p>HW/S: Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so it can be tested in a simulated environment. Examples include “high fidelity” laboratory integration of components.</p> <p>SW: Reliability of software ensemble increases significantly. The basic software components are integrated with reasonably realistic supporting elements so that it can be tested in a simulated environment. Examples include “high fidelity” laboratory integration of software components.</p> <p>System software architecture established. Algorithms run on a processor(s) with characteristics expected in the operational environment. Software releases are “Alpha” versions and configuration control is initiated. Verification, Validation, and Accreditation (VV&A) initiated.</p>
TRL 6	System/subsystem model or prototype demonstration in a relevant environment	<p>HW/S: Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology’s demonstrated readiness. Examples include testing a prototype in a high-fidelity laboratory environment or in a simulated operational environment.</p> <p>SW: Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in software demonstrated readiness. Examples include testing a prototype in a live/virtual experiment or in a simulated operational environment. Algorithms run on processor of the operational environment are integrated with actual external entities. Software releases are “Beta” versions and configuration controlled. Software support structure is in development. VV&A is in process.</p>
TRL 7	System prototype demonstration in an operational environment	<p>HW/S: Prototype near, or at, planned operational system. Represents a major step up from TRL 6, requiring demonstration of an actual system prototype in an operational environment such as an aircraft, vehicle, or space. Examples include testing the prototype in a test bed aircraft.</p> <p>SW: Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment, such as in a command post or air/ground vehicle. Algorithms run on processor of the operational environment are integrated with actual external entities. Software support structure is in place. Software releases are in distinct versions. Frequency and severity of software deficiency reports do not significantly degrade functionality or performance. VV&A completed</p>

TRL	Definition	Description
TRL 8	Actual system completed and qualified through test and demonstration	<p>HW/S: Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation of the system in its intended weapon system to determine if it meets design specifications.</p> <p>SW: Software has been demonstrated to work in its final form and under expected conditions. In most cases, this TRL represents the end of system development. Examples include test and evaluation of the software in its intended system to determine if it meets design specifications. Software releases are production versions and configuration controlled, in a secure environment. Software deficiencies are rapidly resolved through support infrastructure.</p>
TRL 9	Actual system proven through successful mission operations	<p>HW/S: Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation. Examples include using the system under operational mission conditions.</p> <p>SW: Actual application of the software in its final form and under mission conditions, such as those encountered in operational test and evaluation. In almost all cases, this is the end of the last “bug fixing” aspects of the system development. Examples include using the system under operational mission conditions. Software releases are production versions and configuration controlled. Frequency and severity of software deficiencies are at a minimum.</p>

10.7. U.S. Army Software Readiness Level (SRL)

On August 2010, the Carnegie Mellon Software Engineering Institute (SEI) facilitated an Army workshop entitled “Beyond Technology Readiness Levels for Software” at Picatinny Arsenal in New Jersey. The result of the brainstorming is the in which 5 different levels of software readiness have been presented.

The aspect that generates interest for DOGANA is the identification of artifacts to provide evidence of completion.

Table 22 – U.S. Army SRL

SRL	Definition	Completion criteria	Artefacts that could provide evidence of completion
SRL 1	Concept	<ul style="list-style-type: none"> • Features defined • Operations concept defined that represents user expectations • Operational/stakeholder 	<ul style="list-style-type: none"> • Quality attribute scenarios • Stakeholder requirements document • White papers • Algorithms

		<p>requirements are approved</p> <ul style="list-style-type: none"> • Appropriate feasibility studies have been conducted (hardware, software, algorithm) • Mission/capability objectives have been defined for the system • Initial set of quality attributes has been defined • Interfaces are defined (as part of requirements?) 	<ul style="list-style-type: none"> • Initial interface requirements • Software requirements specification (SRS) outline • Use cases • System definitions to show software context (allocation of system functions to hardware, software, humans and relationships among them) / functional architecture • Draft software development plan • Feature impact statement • Concept of operations document • Mission/capability statements
SRL 2	Architecture	<ul style="list-style-type: none"> • Appropriate architecture views (e.g., modular, runtime, deployment) have been defined • Selections of approved software development tools have been made • Architecture evaluations have been conducted • Appropriate standards and protocols have been selected • Interfaces have been refined • Initial CSCIs have been defined • Architecture elements have been allocated • Software test strategy has been defined 	<ul style="list-style-type: none"> • Architecture documented in multiple views • Architecture evaluation results • Interface control specifications • Software test and evaluation plan • System/subsystem design document (SSDD) • Criteria for evaluating prototype • DoD Information Assurance Certification and Accreditation Process (DIACAP) plan/selection of approved software tools
SRL 3	Design/Prototype	<ul style="list-style-type: none"> • Functioning prototype • Software design is defined • CSCIs are refined • Architectural allocations are fleshed out • Results of analyzing prototype(s) • Proof of “reproducibility” (via a functioning configuration management system) • Software test planning is complete 	<ul style="list-style-type: none"> • Test plan/initial draft of test cases • Resource requirements (technical performance measures) • Software design document • Working version description document (VDD) • List of included COTS products • Requirements traceability matrix • Architecture-to-design traceability matrix • Updated SRS • Software WBS • Updated software architecture views • Database schema • Prototype evaluation report • Software development plan
SRL 4	Developed	<ul style="list-style-type: none"> • Unit testing is completed • Software integration is completed • Interfaces have been verified • Software is documented and 	<ul style="list-style-type: none"> • Unit and software integration test results • Functional Configuration Audit/Physical Configuration Audit (FCA/PCA) results

		<p>reviewed</p> <ul style="list-style-type: none"> • Peer reviews are conducted and analyzed • Developed code conforms to documentation • Code is under appropriate configuration management • Quality attributes have been verified • Freeze criteria for code have been defined 	<ul style="list-style-type: none"> • Peer review reports • Updated VDD • Software user's manual • Software quality statement • Metrics map to software quality attributes (technical performance measures [TPM]) and analysis results • Criteria for software freeze • Software Formal Qualification Test (FQT) dry run report • Software test readiness review document • Deployment plan
SRL 5	Ready deployment for	<ul style="list-style-type: none"> • Formal certifications are completed (information assurance [IA], interoperability, safety, authority to operate [ATO], etc.) • System test complete • User acceptance tests completed • Operational tests/assessments are complete • Maintenance strategy defined • Baseline is established • Post-deployment support infrastructure established (field support, tiered support, training, help desk...) 	<ul style="list-style-type: none"> • Certification reports and issued certifications or waivers • Operational test report • VDD • FQT/Software Test Report (STR) • Software user's manual • Software operators manual • Software tech manual • Software sustainment/supportability/suitability plan • User acceptance memo • Deployment plan • Concurrence letter from PM • Validated data loading and installation scripts • Safety and IA checklist (how to use a simple key loader [SKL] to load an encryption key)

11. Annex II: D2.2 Checklist for theoretical assessment deliverables

Table 23 – D2.2 Checklist for theoretical assessment deliverables

	Risk (as described in D1.3 Section 3)	Requirement	Argumentation
Stage 1. Preliminary measures	The research results may have a severe negative impact on the human rights of individuals or groups (e.g. privacy, discrimination, stigmatization)	<p>Risk mitigation, such as</p> <ul style="list-style-type: none"> - a human rights impact assessment - the involvement of human rights experts in the research - training of personnel and/or technological safeguards <p>Risk-assessment</p> <ul style="list-style-type: none"> - details on how the research could affect human rights - details on the measures taken to prevent abuse 	<p>The task is aimed at defining metrics to evaluate existing tools, in terms of their maturity and effectiveness with respect to the possibility to integrate them into DOGANA toolchain.</p> <p>In general, those tools may be able to craft information regarding people through Social Network or other public sources, which potentially impact on human rights. Thus some metrics aimed at evaluating the quality of results could be defined.</p> <p>Nevertheless, metrics themselves will not involve directly people information and they will not impact directly on human rights. Moreover, they will not be applied within this task, but they will be used in following tasks.</p> <p>Even if not foreseen for this task, in order to better understand the functionality of a tool, it might be necessary to test it in a controlled environment, possibly measuring part of the digital footprint of a user, thus obtaining some information (e.g., interests or pictures related to profile). This step might be necessary for example in order to better understand the possible results with the purpose of defining more detailed metrics. In case this evaluation step will be performed during the task, it will be done against ad-hoc crafted fake profiles. Moreover, in any case no</p>

			data retrieved will be used or stored for further analysis.
	The research has the potential to be abused or misused	<p>Risk-assessment</p> <ul style="list-style-type: none"> - details on the measures taken to prevent abuse - if applicable, copies of personnel security clearances 	<p>Metrics, categories, and scoring system that will be defined could allow to retrieve information regarding the actual effectiveness of tools. This information could be used from a malicious user to select potential attack tools.</p> <p>Regarding this threat, metrics themselves will not reveal risky information regarding potential attack tools.</p> <p>Anyway, in order to prevent abuse of following results, additional countermeasure will be adopted during this task.</p> <p>In particular, metrics will be analyzed in terms of possible risks related to abuse of the results of their application. For each metrics with high level of risk, additional recommendation regarding how the results should be evaluated and published will be defined.</p>
Stage 2. Research considerations	The research may have a negative impact on human rights	Research methods for correct interpretation of the research results should be provided	The outcome of the task consists in a set of metrics and a possible method for evaluating existing tools and awareness methods. This outcome have not any negative impact on human rights, thus there is no need to define methods for correct interpretations of the results.
	Confidential Dogana internal information could be disclosed through the research	<p>Caution when publishing or otherwise disseminating those results</p> <p>Compliance with non-disclosure agreements and other (internal) contracts in relation to the research data</p>	Metrics and methods defined within this task are based on publicly documented methodologies and will be applied on existing 3 rd parties' accessible tools. Consequently, the disclosure of the outcome of this task will have not any impact neither on the non-disclosure agreement nor on the intellectual

		Compliance with the technical partner couples relationships	property of DOGANA consortium on the project foreground. On the contrary, the collection of possible external feedbacks may allow improvements of the final outcome.
Stage 3. Post measures	Data loss	Detailed measures on the storage-assessment (including access control) Assessment by the end-users according to WP7 and considering the three different sharing levels	The information included in the deliverable and the concerning related activities are not sensitive or critical. Therefore there is no need to define countermeasure to avoid data loss.
	The research may have a negative impact on human rights	Caution when publishing or otherwise disseminating those results Statement that no data other than the results of the project (software and documentation) will be exported to non-EU Member States	Metrics and methods defined within this task are based on publicly documented methodologies and will be applied on existing 3 rd parties' accessible tools. Therefore, their publication or dissemination will have not any negative impact on human rights.

12. References

- [1] "Attack Surface Analysis Cheat Sheet". Open Web Application Security Project. Available at:
https://www.owasp.org/index.php?title=Attack_Surface_Analysis_Cheat_Sheet&oldid=156006
- [2] M. Rouse, "What is attack vector?," SearchSecurity, 2012. Available at:
<http://searchsecurity.techtarget.com/definition/attack-vector>
- [3] M. Jackson, S. Crouch, and R. Baxter. "Software evaluation: criteria-based assessment" Software Sustainability Institute. Available at:
<http://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf>
- [4] M. Jackson, S. Crouch, and R. Baxter. "software evaluation guide" Software Sustainability Institute. Available at: <http://www.software.ac.uk/software-evaluation-guide>
- [5] "Standard for tackling land degradation takes step closer to publication"
<http://www.iso.org/iso/home.html>
- [6] J. Mara, M. O'Brien, G. Charlton, T. Wegert, and F. Sinton, "Google [Acquires Picasa](#)", Clickz, July 13 2004.
- [7] P. R. La Monica, "[Google to buy YouTube for 1.65 billion](#)". CNN. Retrieved October 9, 2006.
- [8] "[Google To Acquire YouTube for Billion in Stock](#)". [Google](#). Retrieved October 9, 2006.
- [9] "List of defunct social networking websites". Available at:
https://en.wikipedia.org/wiki/List_of_defunct_social_networking_websites
- [10] D. Wallace, "R.I.P. - top 10 failed social media sites" in Social Media, Search Engine Journal, 2013. Available at: <https://www.searchenginejournal.com/r-i-p-top-10-failed-social-media-sites/57554/>
- [11] Manadhata, Pratyusa (2008). An Attack Surface Metric (PDF). Available at:
<http://reports-archive.adm.cs.cmu.edu/anon/2008/CMU-CS-08-152.pdf>
- [12] "Social engineer Toolkit (SET) - security through education". Available at:
<http://www.social-engineer.org/framework/se-tools/computer-based/social-engineer-toolkit-set/>
- [13] "PCI data security standard (PCI DSS): best practices for implementing a security awareness program". Available at:

www.pcisecuritystandards.org/documents/PCI_DSS_V1.0_Best_Practices_for_Implementing_Security_Awareness_Program.pdf

[14] S. Blackmore "Imitation Makes us Human",
<http://www.susanblackmore.co.uk/Chapters/human.pdf>

[15] A. Neupane, M. L. Rahman, N. Saxena, and L. Hirshfield, "A multi-modal Neuro-Physiological study of Phishing detection and Malware warnings," *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15*, 2015.

[16] R. B. Clayton, G. Leshner, and A. Almond, "The extended iSelf: The impact of iPhone separation on Cognition, emotion, and physiology," *Journal of Computer-Mediated Communication*, vol. 20, no. 2, pp. 119–135, Jan. 2015.

[17] B. S. Barn, R. Barn, and J. P. Tan, "Young people and smart phones: An empirical study on information security," *47th Hawaii International Conference on System Sciences*, January 2014.

[18] M. S. Bandor "quantitative methods for software selection and evaluation". Available at: <http://www.sei.cmu.edu/reports/06tn026.pdf>

[19] J. C. Mankins, "Technology readiness assessments: A retrospective," *Acta Astronautica*, vol. 65, no. 9–10, pp. 1216–1223, 2009.

[20] D. McGarvey, D. Olson, J. Savitz, S., Diaz, G. and Thompson, G. (2009) *Department of Homeland Security Science and Technology Readiness Level Calculator*. Available at: http://www.homelandsecurity.org/docs/reports/DHS_ST_RL_Calculator_report20091020.pdf

[21] European Commission Decision C (2015)6776 - General Annexes, Annex G Technology Readiness Levels (TRL) 2015, c. Available at: http://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2016-2017/annexes/h2020-wp1617-annex-ga_en.pdf

[22] "Space systems -- definition of the technology readiness levels (TRLs) and their criteria of assessment,". [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=56064.

[23] P. Graettinger, S. Garcia, J. Sivi, and R. J. Schenk, "Using the technology readiness levels scale to support technology management in the DoD's ATD/STO environments A findings and recommendations report conducted for army CECOM," 2002. Available: http://resources.sei.cmu.edu/asset_files/SpecialReport/2002_003_001_13931.pdf.

[24] C. Albert, S. Garcia-Miller, S. Blanchette, S. Garcia, and S. Miller, "Beyond technology readiness levels for software: U.S. Army workshop report," Publisher: Software Engineering Institute CMU/SEI Report Number: CMU/SEI-2010-TR-044, 2016. Available: <http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=9689>.

